



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## PRIVACY IN PARTICIPATORY SENSING

User-Controlled Privacy-Preserving Solutions for Mobile Sensing Applications

Am Fachbereich Informatik  
der Technischen Universität Darmstadt  
zur Erlangung des akademischen Grades einer  
Doktor-Ingenieurin (Dr.-Ing.)  
genehmigte Dissertationsschrift

von

DIPL.-ING. DELPHINE CHRISTIN

Geboren am 22. Februar 1985 in Fréjus, Frankreich

Erstreferent: Prof. Dr.-Ing. Matthias Hollick  
Korreferent: Prof. Dr. habil. Simone Fischer-Hübner  
Korreferent: Prof. Salil Kanhere (PhD)

Tag der Einreichung: 22. Januar 2013  
Tag der Disputation: 13. Februar 2013

Darmstadt, 2013  
Hochschulkennziffer D17



## ABSTRACT

---

Information and Communication Technology is commonly recognized as one of the key enablers in improving the quality of life in our modern society. By including citizens in the digital world, the efficiency of existing services has already been improved. In a second wave, the technological advances of mobile phones promise to further bridge the gap between physical world and cyberspace. Using sensors embedded in mobile phones, a torrent of data about the physical world can be collected. The inclusion of the gathered data into the digital world contributes to the realization of the vision of so-called smart spaces, ranging from smart homes to smart cities and beyond. These smart spaces can substantially increase the quality of life by leveraging the participation of billions of citizens by, e.g., monitoring traffic congestion and noise pollution. The collection of sensor readings in participatory sensing applications however puts the privacy of the users at risk, as they may reveal sensitive information about themselves, such as the locations they visited. Users aware of such threats may decide to opt out of the application, thus decreasing the quantity and quality of the gathered data. Privacy protection is therefore mandatory to encourage potential contributions.

Most existing privacy-preserving solutions specifically tailored to participatory sensing applications fail to include users in the loop, despite the individual nature of the conception of privacy. They are mainly preconfigured by application administrators and cannot be personalized by users according to their privacy preferences. Moreover, a majority of existing approaches relies on a central entity responsible for the users' privacy protection. In addition to be a single point of failure, users need to entrust them not to disclose sensitive information to unauthorized third parties. In order to address these shortcomings and ultimately foster the users' contributions, we propose three privacy-preserving solutions in which users are in control of their privacy protection and can adapt the underlying mechanisms to their own preferences. Furthermore, the provided privacy protection is independent of the trustworthiness of the application server.

In particular, we present a scheme in which users mutually preserve their privacy by physically exchanging sensor readings, along with the time and location of their collection, during opportunistic meetings. By breaking the association between users' identities and sensor readings, the exchanges prevent curious application administrators from inferring the locations visited by the participants based on the uploaded sensor readings. This scheme, however, relies on the collaboration of all participating users. In order to assess the users' degrees of collaboration, we therefore propose mechanisms based on user ratings that readily identify and quarantine malicious users. Using both schemes, users are hence able to determine the applied exchange strategy based the assessed trust levels of encountered users as well as their individual preferences. As a result, the collected data are obfuscated prior to their upload to the application server.

Furthermore, we propose an innovative scheme based on periodic pseudonyms that allows the application server to assess the trustworthiness of the contributed sensor readings without endangering the users' privacy. In addition to rely on blind signatures, our scheme introduces the concept of reputation cloaking in order to prevent curious application administrators from linking consecutive pseudonyms based on an analysis of their reputation. By applying different proposed cloaking schemes, users can control and balance the inherent trade-off between anonymity protection and loss in reputation according to their personal preferences.

In addition to proposing these three schemes, we investigate the degree of privacy protection achieved by the devised solutions by means of extensive simulations under realistic scenarios and conditions. Besides, we assess the applicability of our contributions in participatory sensing applications by using real-world data traces and through prototypical implementation. Based on our thorough evaluation, we provide guidelines for the parametrization of the proposed schemes by considering both the user and application perspectives.





## ZUSAMMENFASSUNG

---

Informations- und Kommunikationstechnologie wird als einer der wichtigsten Wegbereiter zur Verbesserung der Lebensqualität in unserer modernen Gesellschaft gesehen. Insbesondere die verstärkte Einbindung der Bürger in die digitale Welt führt zu neuartigen Möglichkeiten der Erfassung und Verarbeitung nutzergenerierter Inhalte. Gleichzeitig trägt der technologische Fortschritt bei Mobiltelefonen dazu bei, dass die Grenzen zwischen physikalischer Umgebung und der virtuellen Welt nach und nach verschwimmen. So können mit Hilfe von in heutigen Mobiltelefonen eingebetteten Sensoren große Datenmengen über die Umgebung der Nutzer gesammelt werden. Die Integration dieser Daten in die digitale Welt kann zur Realisierung der Vision intelligenter Umgebungen genutzt werden. Diese Vision reicht von intelligenten Gebäuden bis hin zu ganzen intelligenten Städten und darüber hinaus. Eine Realisierung solcher intelligenter Umgebungen ermöglicht die Beteiligung von Milliarden von Einwohnern, etwa zur Erfassung von Lärmbelastungen, und kann so zu einer signifikanten Verbesserung der Lebensqualität aller führen. Die derartige Sammlung von Messwerten in den so genannten partizipativen Sensornetzen kann jedoch die Privatsphäre der Nutzer gefährden, da durch sie sensible Informationen über die Nutzer, z.B. besuchte Orte, preisgegeben werden können. Die Kenntnis solcher Bedrohungen der Privatsphäre kann die Teilnahmebereitschaft der Nutzer negativ beeinflussen und damit sowohl Quantität als auch Qualität der gesammelten Daten mindern. Es ist daher ein adäquater Schutz der Privatsphäre vonnöten, um entsprechende Nutzerbeiträge zu fördern.

In den meisten existierenden partizipativen Sensornetzen werden die Nutzer in Entscheidungen über den Schutz ihrer Privatsphäre nicht einbezogen, trotz des nachweislich individuellen Charakters des Konzepts der Privatsphäre. Stattdessen werden die Entscheidungen hauptsächlich von Anwendungsadministratoren getroffen und können nicht von Benutzern personalisiert oder ihren Präferenzen entsprechend konfiguriert werden. Zudem beruht die Mehrheit vorhandener Ansätze auf einer zentralen Instanz, die für den Schutz der Privatsphäre der Nutzer verantwortlich ist. Diese zentrale Instanz stellt allerdings nicht nur einen "single point of failure" dar, die Nutzer müssen darüber hinaus auch darauf vertrauen, dass von dieser Instanz keine sensiblen Nutzerinformationen an unbefugte Dritte weitergegeben werden.

Um diese Einschränkungen aufzuheben und somit letztendlich die Beteiligung von Nutzern in partizipativen Sensornetzen zu steigern, werden im Rahmen dieser Arbeit drei Lösungen vorgeschlagen, mit welchen die Nutzer ihre Privatsphäre wirksam schützen und die zugrunde liegenden Mechanismen an ihre eigenen Wünsche anpassen können. Darüber hinaus ist der erzielte Schutz der Privatsphäre in den vorgestellten Verfahren unabhängig von der Vertrauenswürdigkeit eines Anwendungsservers. Zunächst wird eine Lösung zur Verschleierung von Bewegungsmustern vorgestellt, in der Nutzer kollaborieren, um ihre Privatsphäre durch den opportunistischen Austausch zeitlich und örtlich annotierter Sensorwerte zu schützen. Durch diesen Vorgang wird die Verbindung zwischen Nutzeridentitäten und gesammelten Sensorwerten aufgehoben, sodass auch Anwendungsadministratoren auf Basis zur Verfügung gestellter Sensorwerte nicht mehr auf besuchte Orte schließen können.

Die Effektivität der vorgeschlagenen Lösung hängt von der Zusammenarbeit aller beteiligten Nutzer ab. Entsprechend werden zusätzliche Mechanismen vorgestellt, mit denen der Kollaborationsgrad von Nutzern bewertet werden kann und bössartige Nutzer identifiziert und gezielt ausgeschlossen werden können. Durch die Verwendung dieser Mechanismen können Nutzer ihre Strategien zum Datenaustausch anhand von Vertrauensbewertungen und ihren individuellen Präferenzen festlegen.

Schließlich wird im Rahmen dieser Arbeit ein Verfahren vorgestellt, das es einem Anwendungsserver ermöglicht, auf Basis periodischer Pseudonyme die Vertrauenswürdigkeit der übertragenen Sensorwerte ohne Gefährdung der Privatsphäre der Nutzer zu beurteilen. Hierzu wird das Konzept der "Reputationsverschleierung" eingeführt, um zu verhindern,

dass Anwendungsadministratoren aufeinanderfolgende Pseudonyme durch eine Analyse ihrer Reputationswerte in Verbindung zueinander bringen können. Durch die Verwendung verschiedener Methoden zur Reputationsverschleierung können die Nutzer selbst das Verhältnis zwischen Anonymität und dem Reputationsverlust, der mit einer Reputationsverschleierung einhergeht, auf Basis ihrer persönlichen Präferenzen festlegen.

Für alle vorgestellten Lösungen wird der erreichte Grad des Schutzes der Privatsphäre mittels umfangreicher Simulationen anhand von realistischen Szenarien untersucht. Außerdem wird die Anwendbarkeit der Lösungen durch Anwendung realer Datensätze und prototypischer Implementierungen evaluiert.

## ACKNOWLEDGMENTS

---

My first thanks go to Prof. Matthias Hollick for his supervision and constant support during my doctoral studies. I also thank Prof. Simone Fischer-Hübner for having inspired my studies since our first meeting in 2009 and having accepted to be my co-referee. I further thank Prof. Salil S. Kanhere for his valuable feedback and his hospitality during my different research visits at the University of New South Wales, Sydney, Australia. Additionally, I would like to thank Prof. Melanie Volkamer, Prof. Ralf Steinmetz, and Prof. Max Mühlhäuser for their contributions as members of my committee.

I would like to express my gratitude to all my co-authors for their support and constructive discussions, especially Christian Roßkopf, Leonardo Martucci, and Nicolas Repp. My thanks also go to my students for their contributions and our enjoyable discussions. Thanks to my colleagues at the *Secure Mobile Networking Lab* as well as the *Center for Advanced Security Research Darmstadt* for the good working atmosphere and opening my mind to further research directions. In particular, my thanks go to Doris Müller and Alexander Oberle for their endless support.

I am grateful to past and present members of the *Multimedia Communications Lab* for the interesting discussions across the years, especially Sonja Bergsträßer, Doreen Böhnstedt, Christian Gottron, Jan Hansen, Monika Jayme, Frank Jöst, André König, Sabine Kräh, Robert Lokaiczky, André Miede, Silvia Rödelberger, Gisela Scholz-Schmidt, Karola Schork-Jakobi, and Stephan Tittel. Further special thanks go to Verena and Sebastian Zöller.

Additionally, I would like to thank my parents, grandparents and family for their permanent encouragement. Finally, I would like to express my sincere gratitude to you, Andreas, for sharing this journey with me.

*Darmstadt, 2013*

Delphine



## CONTENTS

---

1	INTRODUCTION . . . . .	1
1.1	Motivation . . . . .	1
1.2	Goals . . . . .	2
1.3	Methodology . . . . .	3
1.4	Contributions . . . . .	3
1.4.1	Path Jumblng: A Privacy-preserving Collaborative Scheme for Path Hiding . . . . .	3
1.4.2	TrustMeter: A Trust Assessment Scheme for Collaborative Path Hiding . . . . .	4
1.4.3	IncogniSense: An Anonymity-preserving Reputation Framework . . . . .	4
1.5	Outline . . . . .	5
2	FOUNDATIONS . . . . .	7
2.1	Participatory Sensing Applications . . . . .	7
2.1.1	People-centric Sensing Applications . . . . .	7
2.1.2	Environment-centric Sensing Applications . . . . .	10
2.2	System Model . . . . .	12
2.2.1	Stakeholders . . . . .	12
2.2.2	Architecture . . . . .	12
2.3	Privacy . . . . .	15
2.3.1	Definition . . . . .	15
2.3.2	Privacy Analysis . . . . .	16
2.3.3	Privacy Threats . . . . .	17
2.4	Summary . . . . .	19
3	RELATED WORK . . . . .	21
3.1	Tailored Sensing and User Preferences . . . . .	21
3.2	Privacy-preserving Task Distribution . . . . .	23
3.3	Anonymous and Privacy-preserving Data Reporting . . . . .	24
3.3.1	Pseudonymity . . . . .	25
3.3.2	Spatial Cloaking . . . . .	25
3.3.3	Data Perturbation . . . . .	26
3.3.4	Hiding Sensitive Locations . . . . .	27
3.3.5	Data Aggregation . . . . .	27
3.4	Privacy-aware Data Processing . . . . .	28
3.5	Review, Deletion, Storage, and Retention of Data . . . . .	28
3.6	Access Control and Audit . . . . .	29
3.7	Privacy-aware Data Query . . . . .	29
3.8	Summary . . . . .	30
4	PROBLEM STATEMENT . . . . .	31
4.1	Application-controlled to User-controlled Privacy-preserving Schemes . . . . .	31
4.2	Centralized to Decentralized Privacy-preserving Solutions . . . . .	32
4.3	Summary . . . . .	32
5	PATH JUMBLING: A PRIVACY-PRESERVING COLLABORATIVE SCHEME FOR PATH HIDING . . . . .	35
5.1	System and Threat Models . . . . .	35
5.1.1	System Model . . . . .	35
5.1.2	Threat Model . . . . .	36
5.2	The Path Jumblng Concept . . . . .	37
5.2.1	Overview . . . . .	37

5.2.2	Exchange Strategies . . . . .	37
5.2.3	Reporting Strategies . . . . .	40
5.3	Performance in Exchanging and Reporting Triplets . . . . .	42
5.3.1	Simulation Setup and Method . . . . .	42
5.3.2	Results . . . . .	43
5.4	Privacy Protection . . . . .	48
5.4.1	Simulation Setup and Method . . . . .	48
5.4.2	Evaluation Metric . . . . .	52
5.4.3	Results . . . . .	54
5.4.4	Evaluation Summary . . . . .	63
5.5	Related Work . . . . .	63
5.6	Summary . . . . .	64
6	TRUSTMETER: A TRUST ASSESSMENT SCHEME FOR COLLABORATIVE PATH HID- ING . . . . .	65
6.1	System and Threat Models . . . . .	65
6.1.1	System Model . . . . .	65
6.1.2	Threat Model . . . . .	66
6.2	TrustMeter Architecture . . . . .	66
6.2.1	Overview . . . . .	66
6.2.2	Underlying Mechanisms . . . . .	67
6.3	Robustness against Threats to Reputation and Privacy . . . . .	69
6.3.1	Manipulation of Reputation . . . . .	69
6.3.2	Replay and Sybil Attacks . . . . .	70
6.4	Identification of Malicious Users . . . . .	70
6.4.1	Simulation Setup and Method . . . . .	70
6.4.2	Framework Parameterization . . . . .	71
6.4.3	Identification of Droppers . . . . .	76
6.4.4	Spammers . . . . .	82
6.4.5	Evaluation Summary . . . . .	84
6.5	Traffic Overhead . . . . .	86
6.6	Related Work . . . . .	87
6.7	Summary . . . . .	88
7	INCOGNISENSE: AN ANONYMITY-PRESERVING REPUTATION FRAMEWORK . . .	89
7.1	Assumptions and Threat Model . . . . .	89
7.1.1	Threats to Reputation . . . . .	89
7.1.2	Threats to Anonymity . . . . .	89
7.2	The IncogniSense Framework . . . . .	90
7.2.1	Overview and Underlying Mechanisms . . . . .	90
7.2.2	Cloaking Mechanisms . . . . .	91
7.2.3	Comparison with the “Reputation using Pseudonyms” Framework . .	93
7.3	Robustness against Threats to Reputation . . . . .	93
7.3.1	Sybil Attacks . . . . .	94
7.3.2	Replay Attacks . . . . .	94
7.3.3	Manipulation of Reputation Accounts . . . . .	94
7.3.4	Reporting of Falsified Sensor Readings . . . . .	94
7.4	Anonymity Protection . . . . .	94
7.4.1	Transient Reputation Scores . . . . .	95
7.4.2	Permanent Reputation Scores . . . . .	102
7.5	Empirical Evaluation . . . . .	109
7.6	Related Work . . . . .	110
7.7	Summary . . . . .	111
8	CONCLUSIONS AND OUTLOOK . . . . .	113

8.1	Summary and Conclusions . . . . .	113
8.2	Outlook . . . . .	114
	BIBLIOGRAPHY . . . . .	117
	LIST OF ACRONYMS . . . . .	129
A	APPENDIX . . . . .	131
A.1	Blind Signatures, RSA Signatures and Blind RSA Signatures . . . . .	131
A.2	Generation of Pseudonyms using RSA Signatures . . . . .	131
A.3	Generation of Reputation Tokens using RSA Signatures . . . . .	132
B	CURRICULUM VITÆ . . . . .	133
C	AUTHOR'S PUBLICATIONS . . . . .	135
D	ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG . . . . .	139





## INTRODUCTION

---

The beginning is the most  
important part of any work.

---

Plato

### 1.1 MOTIVATION

Mobile phones have been riding the wave of Moore's Law with rapid improvements in processing power, embedded sensors, storage capacities, and network data rates. The mobile phones of today have evolved from merely being phones to full-fledged computing, sensing, and communication devices. It is thus hardly surprising that almost 6 billion people globally have access to mobile phones [Int11]. These advances in mobile phone technology, coupled with their ubiquity have paved the way for an exciting new paradigm for accomplishing large-scale sensing, known in literature as *participatory sensing*<sup>1</sup> [BEH<sup>+</sup>06, CEL<sup>+</sup>06]. The key idea behind participatory sensing is to empower ordinary citizens to collect and share sensed data from their surrounding environment using their mobile phones. Mobile phones, though not built specifically for sensing, can in fact readily function as sophisticated sensors. The cameras on mobile phones can be used as video and image sensors. The microphone on the mobile phone, when it is not used for voice conversations, can double up as an acoustic sensor. The embedded Global Positioning System (GPS) receivers on the phone can provide location information. Other embedded sensors such as gyroscopes, accelerometers, and proximity sensors can collectively be used to estimate useful contextual information (e.g., if the user is walking or traveling on a bicycle). Further, additional sensors can be easily interfaced with the phone via Bluetooth or wired connections, e.g., air pollution or biometric sensors. The collected sensor readings can then be transferred to the application using standard communication infrastructures, such as Wi-Fi or 3rd Generation (3G) connectivity.

A plethora of novel and exciting participatory sensing applications have emerged in recent years. CarTel [HBZ<sup>+</sup>06] is a system that uses mobile phones carried in vehicles to collect information about traffic, quality of en-route Wi-Fi access points, and potholes on the road. Micro-Blog [GLC<sup>+</sup>08] is an architecture which allows users to share multimedia blogs enhanced with inputs from other physical sensors of the mobile phone. Other applications of participatory sensing include the collection and sharing of information about urban air [PHG07] and noise pollution [RCK<sup>+</sup>10], cyclist experiences [EML<sup>+</sup>09], diets [RPH<sup>+</sup>07], or consumer pricing information in offline markets [DKCB08]. A typical participatory sensing application operates in a centralized fashion, i.e., the sensor data collected by the phones of volunteers are reported (using wireless data communications) to a central server for processing, as illustrated in Figure 1. The sensing tasks on the phones can be triggered manually, automatically, or based on the current context. On the server, the data are analyzed and made available in various forms, such as graphical representations or maps showing the sensing results at individual and/or community scale. Simultaneously, the results may be displayed locally on the carriers' mobile phones or accessed by the larger public through web-portals depending on the application needs.

---

<sup>1</sup> Without loss of generality, we use the generic term *participatory sensing* to designate applications using mobile phones as sensors (or as data sink for interfaced sensors) where participants voluntarily contribute sensor data for their own benefit and/or the benefit of the community. The notion of participatory sensing therefore includes *mobiscopes* [AAB<sup>+</sup>07] and *opportunistic sensing* [CEL<sup>+</sup>06], and also covers specific terminologies focusing on particular monitoring subjects, such as *urban sensing* [CEL<sup>+</sup>06], *participatory urbanism* [PHG07], *citizen sensing* [BEH<sup>+</sup>06], *people-centric sensing* [CEL<sup>+</sup>06, CEL<sup>+</sup>08], and *community sensing* [KHKZ08].

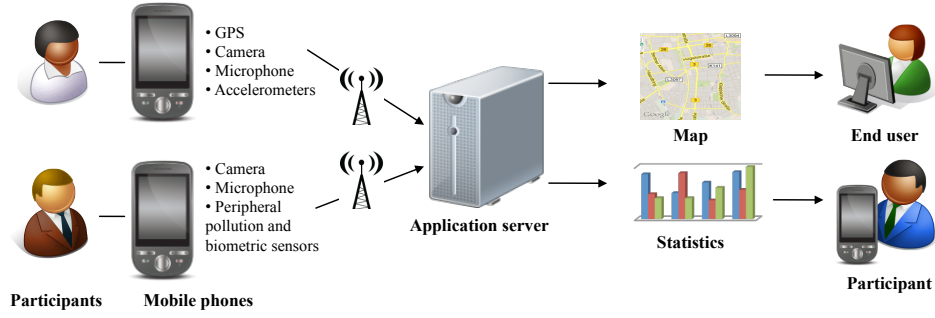


Figure 1: Architectural overview of a typical participatory sensing application

Participatory sensing therefore offers a number of advantages over traditional sensor networks which entails deploying a large number of static wireless sensor devices, particularly in urban areas [CH11]. First, since participatory sensing leverages existing sensing (mobile phones) and communication (cellular or Wi-Fi) infrastructure, the deployment costs are virtually zero. Second, the inherent mobility of the phone carriers provides unprecedented spatiotemporal coverage and also makes it possible to observe unpredictable events. Third, using mobile phones as sensors intrinsically affords economies of scale. Fourth, the widespread availability of software development tools for mobile phone platforms and established distribution channels in the form of App stores (e.g., Apple iTunes Store [App13] and Google Play Store [Goo13]) makes application development and deployment relatively easy. Finally, by including people in the sensing loop, it is now possible to design applications that can dramatically improve the day-to-day lives of individuals and communities.

Current participatory sensing applications primarily focus on the collection of data on a large scale. Without any suitable protection mechanism, the mobile phones are transformed into miniature spies, possibly revealing private information about their owners. Possible intrusions into a user's privacy include the recording of intimate discussions, taking photographs of private scenes, or tracing a user's path and monitoring the locations she has visited. In the face of such significant threats to privacy, persuading users to willingly contribute data may be difficult. However, participatory sensing exclusively depends on user-provided data. Hence, a large number of participants is required. The users' reluctance to contribute would diminish the impact and relevance of sensing campaigns deployed at large scale, as well as limiting the benefits to the users. Mechanisms to preserve user privacy are hence mandatory in order to both encounter the risk that a user's privacy might be compromised and encourage contribution as well as gain widespread acceptance among participants.

## 1.2 GOALS

As highlighted in our survey [CRKH11], most existing privacy-preserving mechanisms tailored to the requirements of participatory sensing applications do not involve the users in any decisions related to the protection of their privacy. The users can therefore not apply any control on their data and configure the proposed solutions according to their own privacy preferences, despite the individual nature of the conception of privacy [Wes67]. Indeed, the conception of privacy is influenced by multiple factors, such as the user's context [FHHK<sup>+</sup>11], age, gender, and cultural background [LS93, GHW<sup>+</sup>11]. Moreover, most schemes rely on a central entity, either the application server or a trusted third party, to which users entrust their sensor readings (along with the location coordinates and the time of their collection). Users are hence obliged to trust these entities not to violate their privacy by disclosing these data to unauthorized parties. Furthermore, these entities must protect user privacy by applying appropriate schemes to the reported data, since users lose the control over them once they are uploaded.

Driven by these shortcomings, the key objective of this thesis is to investigate novel privacy-preserving approaches that are customizable by the users and reduce the users' dependency

on central entities to protect their privacy. In particular, our objective can be decomposed into the following three research goals pursued throughout this thesis.

- ▷ To investigate how to give more control to the users over their data and decouple the privacy-protection schemes from any central entity, while simultaneously maintaining the normal function of the underlying participatory sensing applications.
- ▷ To design and build privacy-preserving schemes that can be configured by the users according to their preferences in terms of privacy.
- ▷ To measure the impact of the designed schemes on the function and associated parameters of typical participatory sensing scenarios and evaluate the provided privacy protection under realistic conditions.

### 1.3 METHODOLOGY

In order to achieve the aforementioned research goals, we have first conducted a literature study and identified the shortcomings of existing solutions. We have next designed novel schemes that address the identified shortcomings based on a systematic and thorough analysis of the design space. We have implemented the designed schemes and evaluated them by means of extensive simulations. We have chosen to evaluate the performance of our schemes based on simulations, as they allow a better exploration of the space of parameter values as compared to analytical modeling and measurement. Moreover, due to the lack of available participatory sensing systems, it was not possible to integrate and measure the performance of our developed schemes in real-world applications. We have, however, adopted realistic conditions as basis for our simulations in order to model a real-world participatory sensing system as faithfully as possible. For example, this includes the utilization of a GPS dataset collected by users. We have chosen not to analytically model our schemes, as this would have required us to introduce additional simplifications and assumptions. Additionally, we intended to recreate the same conditions as in existing participatory sensing applications, i.e., the reporting of sensor readings collected by the users to an application server. We have furthermore implemented a proof-of-concept of one of our designed solutions on state-of-the-art mobile phones in order to demonstrate its feasibility under real-world conditions. We have finally analyzed the results of the performance evaluation of our schemes and discussed them in this thesis, before drawing the corresponding conclusions.

### 1.4 CONTRIBUTIONS

In order to overcome the identified shortcomings, we make the following contributions that follow the line defined by our research goals and can be summarized as follows.

#### 1.4.1 *Path Jumbling: A Privacy-preserving Collaborative Scheme for Path Hiding*

We present an application-agnostic, decentralized, and collaborative mechanism to preserve location privacy during the collection of sensor readings and more particularly, the paths followed by the users. As most sensor readings are geotagged, we propose to exchange the collected sensor readings between users in physical proximity. By exchanging their sensor readings, users jumble their paths; the prior path of one participant becomes that of another participant and vice versa. The repetition of the jumbling process at each encounter results in the construction of paths composed of concatenated subpaths from multiple users. As a result, the sensor readings, which are reported to the server by the users, do not disclose the actual paths, but instead a path jumbled with other users. Based on a design space analysis, we select and implement different strategies for exchanging and reporting the sensor readings. These strategies can be adjusted to the desired level of privacy and reporting latency by the users. We analyze the impact of these strategies on the incurred overhead, reached jumbling

degree, and the distance between original and jumbled paths using real-world GPS traces of mobile users. We quantify the provided privacy protection by means of extensive simulations, in which malicious application administrators attempt to identify jumbled triplets from original ones based on the associated spatiotemporal information. The results demonstrate the feasibility and efficacy of our scheme, which prevents malicious application administrators from recognizing most of the locations actually visited by the users.

#### 1.4.2 *TrustMeter: A Trust Assessment Scheme for Collaborative Path Hiding*

As the above concept of path jumbling relies on the collaboration of all participating users, we further propose TrustMeter, an application-agnostic scheme that assesses the behavior of the users contributing to the collaborative path hiding process. It attributes a trust level to each user based on statistics provided by the users about past exchanges. Malicious users can be clearly identified based on their low trust ratings and subsequently excluded from the data jumbling process. Ultimately, users can leverage these ratings to select the appropriate exchange strategy to apply when meeting other users based on their preferences. At the same time, the ratings provide as little information as possible about the exchanges in order to preserve the provided privacy protection. We implement the TrustMeter scheme and investigate its performance in identifying malicious users in a realistic application scenario. By means of extensive simulations, we analyze various attack scenarios and show that TrustMeter identifies most malicious users in the studied scenarios. We further analyze the robustness of TrustMeter against threats to reputation and quantify the overheads introduced by our approach.

#### 1.4.3 *IncogniSense: An Anonymity-preserving Reputation Framework*

Participatory sensing applications are exposed to incorrect contributions due to their inherent open nature [HKH10a]. For example, participants may inadvertently position the phone in an undesirable position while collecting sensor readings, or deliberately contribute bad data. Both behaviors result in erroneous contributions, which need to be identified and eliminated to ensure the reliability of the computed summaries. Reputation systems assign reputation scores to the users based on the quality of their contributions and then, use these scores to weed out bad contributions. However, such systems need to observe the contributions made by each device for an extended period of time to compute the reputation. They hence require linkability across multiple contributions from the same device. Attackers, especially malicious application administrators, can exploit these links to de-anonymize the volunteers and compromise their privacy, since the sensor readings usually include spatiotemporal meta-data [CRKH11]. We thus propose IncogniSense, an anonymity-preserving reputation framework based on blind signatures [Cha83], that addresses this inherent conflict between privacy and reputation requirements and is agnostic to both the reputation assignment algorithm and the application. In IncogniSense, each user picks a new pseudonym for each time period, which is used to report sensor readings. Before the next period starts, the user transfers the reputation score associated with his current pseudonym to his next pseudonym. This allows the user to conserve his reputation across multiple periods, while limiting associations between his contributions to a unique period. IncogniSense cloaks the reputation to be transferred to prevent malicious application administrators from linking multiple pseudonyms. As reputation cloaking has an inherent trade-off between anonymity protection and loss in reputation, we explore this design space by undertaking a detailed simulation-based analysis of several cloaking mechanisms and we especially study their resilience against linking attacks. Based on this analysis, we provide guidelines for the choice of the appropriate cloaking scheme in accordance with the application requirements and the preferences of the users. We conduct a thorough threat analysis showing the robustness of IncogniSense against reputation corruption and show the feasibility of our approach in a practical setup.

## 1.5 OUTLINE

The remainder of this thesis is structured as follows:

Chapter 2 “*Foundations*” presents a detailed introduction to participatory sensing. We introduce the notion of privacy and highlight threats to privacy raised in this particular scenario.

Chapter 3 “*Related Work*” summarizes related work to the field of countermeasures to threats to privacy in participatory sensing applications and highlights shortcomings of existing privacy-preserving solutions.

Chapter 4 “*Problem Statement*” refines the identified shortcomings into research questions addressed throughout this thesis.

Chapter 5 “*Path Jumbling: A Privacy-preserving Collaborative Scheme for Path Hiding*” presents the design and the evaluation of our scheme based on physical exchanges of sensor readings between users in an opportunistic fashion.

Chapter 6 “*TrustMeter: A Trust Assessment Scheme for Collaborative Path Hiding*” supplements the path jumbling approach by providing a scheme that assesses the behavior of the users in the proposed collaborative privacy-preserving scheme based on peer-based ratings.

Chapter 7 “*IncogniSense: An Anonymity-preserving Reputation Framework*” details our approach that addresses the inherent conflict between reputation and privacy, while not trusting the central entities in presence.

Chapter 8 “*Conclusions and Outlook*” concludes this thesis with a summary of our contributions and possible future extensions.



---

If you have built castles in the air, your work  
need not be lost; that is where they should  
be. Now put the foundations under them.

---

H. D. Thoreau

This chapter presents an introduction to participatory sensing based on our previous publication [CRKH11]. In Section 2.1, we first provide an overview of more than 30 representative applications. In this overview, we specially examine the different sensing modalities of each application. Based on this overview, we derive a general system model and present the resulting architecture components as well as the involved stakeholders in Section 2.2. We finally discuss the notion of privacy in participatory sensing systems. This includes the introduction of our definition of privacy tailored to the specificities of participatory sensing systems, in order to address the lack of a common understanding of privacy [New95]. We also conduct a privacy analysis in order to determine actors and processes that represent threats to the privacy of the participants of participatory sensing applications, before highlighting possible consequences resulting from the disclosure of sensitive information.

## 2.1 PARTICIPATORY SENSING APPLICATIONS

The emergence of the participatory sensing paradigm has resulted in a broad range of novel sensing applications, which can be categorized as either *people-centric* or *environment-centric* sensing. People-centric applications mainly focus on documenting activities (e.g., sport experiences) and understanding the behavior (e.g., eating disorders) of individuals. In contrast, environment-centric sensing applications collect environmental parameters (e.g., air quality or noise pollution). As many of the applications make use of the same sensing modalities, we confine our discussion to a selection of representative applications and illustrate the varied usage models of participatory sensing in this thesis.

### 2.1.1 *People-centric Sensing Applications*

People-centric sensing uses the sensor devices integrated in mobile phones to collect data about the user. We discuss a representative selection of existing people-centric participatory sensing applications and analyze the sensing modalities used.

#### *Personal Health Monitoring*

In personal health monitoring, mobile phones are used to monitor the physiological state and health of patients/users using embedded or external sensors (e.g., wearable accelerometers or air pollution sensors). For example, *DietSense* [RPH<sup>+</sup>07] assists participants who want to lose weight by documenting their dietary choices through images and sound samples. The mobile phones are worn on necklaces and automatically take images of the dishes in front of the participants. The images document the participants' food selections and allow for an estimation of the food weight and waste on the plates. Moreover, the mobile phones capture the participants' context during their meals by recording time of day, location, and sound samples to infer potential relationships between the participants' behavior and their context (e.g., having lunch in a restaurant or eating chips late at night on the sofa). All captured data are uploaded to a personal repository, where the participants can review them in order to select or discard the information to be shared with their doctors and nutritionists.

A system to monitor pediatric obesity through multimodal activity detection is presented in [AMM<sup>+</sup>08]. The system is based on a heterogeneous wireless body-area network which employs sensors for heart frequency, acceleration, electrocardiography, blood oxygen saturation, and the user's galvanic skin response. All sensor data are tagged with location information, and optionally with audio and video tags. Adult obesity often results from an imbalance between calorie intake and calorie expenditure. The *BALANCE* system [DAC<sup>+</sup>09] combines an intuitive entry form for calorie intake with a body-area sensor which caters for activity classification. The system relies on the analysis of acceleration patterns to classify the participants' activities, e.g., sitting, running, walking, or bicycling. By correlating activity types and their corresponding durations, the participants' calorie expenditures are estimated. Activity and calorie monitoring is also combined in the *Jog Falls* project [NBB<sup>+</sup>10], which is based on the combination of body-area sensors (acceleration and heart rate) with a simple interface for entering calorie intake. Both functionalities are integrated within a mobile phone application. Complemented by separate blood pressure and weight measurements, participants and their nutritionists are notified of the overall achievements with regard to their diet and targeted weight loss.

The *HealthSense* project [SDAB08] targets the automated detection of health-related events that cannot be directly observed by current sensor technology, like tow conditions, pain, or depression. Acceleration data are being used in conjunction with machine learning approaches to detect correlations, e.g., to diagnose itching as the source for a user's scratching motions. *MobAsthma* [KBRL09] monitors the asthma condition of the patients and their exposure to pollution. A peak flow meter and a pollution sensor are interfaced to the mobile phone via a Bluetooth connection, and measure both the volume of air inhaled and expired by the patients as well as the airborne particle concentration. These measurements are coupled with the patients' location and made available to allergists and asthma specialists to investigate the personal relationships between asthma attacks and exposure to air pollution. In addition to investigation purposes, *MobAsthma* can detect asthma attacks in early stages and autonomously alert remote medical staff.

Mobile phones can also be applied to remotely monitor the activity and posture of patients (e.g., elderly people living alone) using peripheral or the embedded accelerometers [GFH09]. For example, medical staff can gather the physical condition of elderly people by analyzing the temporal repartition of their postures among, e.g., sitting, standing, or lying. The granularity and accuracy of the activity recognition depend on the amount and position of the accelerometers (worn, e.g., on the hips, or in front/back pockets). Similarly to *MobAsthma*, medical staff may be directly alerted via the mobile phones in case of abnormal behavior or when participants fall with *SenSay* [SSF<sup>+</sup>03].

#### *Calculating Environmental Impact and Exposure to Particles*

*PEIR* (Personal Environmental Impact Report) is a system that allows participants to use their mobile phone to determine their exposure to environmental pollutants [MRS<sup>+</sup>09]. A sensing module installed on the phone determines the current location of the user as well as information about the currently used mode of transportation (e.g., bus vs. car), and transfers this information to a central server. In return, the server provides the participants with information about the environmental impact of their traveling in terms of carbon and particle emissions. Additionally, the server estimates the participants' exposure to particle emissions generated by other vehicles and the distance to fast food restaurants while commuting. The latter may be useful for health conscious participants who may want to avoid the temptation of stopping by such restaurants. The mode of transport is inferred using accelerometer readings, while the route travelled is extracted from the captured location traces. Additional input parameters and models are considered for determining the environmental factors, such as weather conditions collected by weather stations, road traffic flow models, and vehicle emission models.



### *Monitoring and Documenting Sport Experiences*

The *BikeNet* [EML<sup>+</sup>07, EML<sup>+</sup>09], *Biketastic* [Shio9], and *SkiScape* [EC06, ELM<sup>+</sup>06] projects monitor the sport experiences of the participants. Both *BikeNet* and *Biketastic* document the bicycling experiences of the participants. *BikeNet* draws a fine-grained portrait of the cyclist by measuring his current location, speed, burnt calories, and galvanic skin response. Multiple peripheral sensors are used to obtain this information: Microphone, magnetometer, pedal speed sensor, inclinometer, lateral tilt, stress monitor, speedometer/odometer, and a sensor for CO<sub>2</sub> concentration. The peripheral sensors form a body area network and interact with the mobile phone over a wireless connection. In comparison, *Biketastic* concentrates on the road conditions, including the roughness of the road and the noise level along the road captured by on-board accelerometers and microphones, respectively. The captured data can be reviewed by the cyclists themselves, but can also be merged with other participants' data or combined with additional parameters, such as air quality and traffic properties, in order to construct complete maps for the cycling community.

In contrast, *SkiScape* focuses on winter sports and is deployed in ski resorts. Peripheral sensors (temperature sensor, accelerometers, and microphones) are attached either to skis or personal equipment to measure the body temperature, the maximum acceleration, and the travelled distance. Besides the sensors bound to the user, several static nodes are located along the trail and cater for the localization of the user within the ski resort. Using *SkiScape*, the participants can document their traces, locate their friends, and select lifts depending on the queue length. Simultaneously, the manager can optimize the maintenance operations, and emergency staff can easily localize users in case of accidents.

### *Enhancing Social Media*

A large pool of applications utilizes data captured by sensors to enrich the contents shared in social media, such as blogs, social networks, or virtual worlds. With *Micro-Blog* [GLC<sup>+</sup>08], participants can create geotagged blog entries and enhance them with multimedia information (e.g., audio records, pictures, accelerometer data, or Wi-Fi coverage) captured via their mobile phones. The created entries are then uploaded to a server, which may position them at their capture location on a global map accessible by the public. Users can browse the entries displayed on the map to find information about particular points of interests (e.g., audio reviews about restaurants, or pictures of the nearest beaches). If the information required is not contained in existing entries, the participants can send queries (e.g., "how is the Wi-Fi coverage near this beach?") to the server, which relays them to mobile phones currently located in this area.

Similarly, *CenceMe* [MLF<sup>+</sup>08, MML<sup>+</sup>08] integrates virtual representations of a participants' current states and context in social networks and virtual worlds. Based on multimodal information (acceleration, audio samples, pictures, neighboring devices, and location) captured by the mobile phone, context information is inferred in various dimensions, including the user's mood, location, and habits, as well as information about the currently performed activity and the environment. The inferred information is then posted as status message in social networks or translated into the virtual representation of participants in virtual worlds.

### *Price Auditing*

*LiveCompare* [DC09] and *PetrolWatch* [DKCB08] facilitate price comparisons of grocery products and fuel at different locations. Instead of manually reporting the prices, the participants use their mobile phones to take pictures of the displayed prices. In *LiveCompare*, the participants only need to take pictures of a product's price tag and its barcode. The barcode is decoded into a textual representation on the mobile phone, and transferred to the server along with capture time, location information, and the picture displaying the current price. In order to compare prices, users can search for products in the application, which then retrieves the corresponding price reports, selects the stores in proximity of the user's current location and displays the pictures of the corresponding price tags.

In comparison, the price collection process is not only simplified in PetrolWatch, but also automated. Each mobile phone is mounted on the passenger seat of a car and faces the road to automatically photograph fuel price boards (using GPS and Geographic Information System (GIS) data) when the vehicle approaches service stations. The pictures are then uploaded to a central entity, which is responsible for image processing and price extraction. The brand of the service station is first inferred from the capture location in order to reduce the image processing complexity, as price boards of different brands differ in colors and dimensions. Assisted by this information, computer vision algorithms extract the fuel prices, which are then uploaded to the database. Users can query the system to determine the cheapest fuel that is available in their area of interest.

### 2.1.2 *Environment-centric Sensing Applications*

In environment-centric scenarios, the mobile phones capture information via their embedded sensors and additional peripheral sensors about the surroundings of the participants. In contrast to most people-centric sensing scenarios, the captured data are mainly exploited at a community scale, e.g., to monitor the evolution of environmental parameters like air quality, thermal columns, noise, road and traffic conditions in cities, or to detect socially interesting events.

#### *Air Quality Monitoring*

In *Haze Watch* [CYCS10], mobile phones were interfaced to external pollution sensors in order to measure the concentration of carbon monoxide, ozone, sulphur dioxide, and nitrogen dioxide concentration in the air. In comparison to meteorological stations, the mobile phones may collect less accurate measurements. However, their inherent mobility provides large spatial coverage and allows them to observe unpredictable events (e.g., accidental pollution), which can seldom be detected by static stations. The mobile phones can thus complement static high-fidelity data captured by traditional meteorological stations by providing finer-grained readings. In addition to pollution measurements, the mobile phones can capture temperature and wind speed, such as shown in *PollutionSpy* [KBRL09] and by [PHG07]. The timestamped and geotagged measurements are then uploaded to a server to build maps accessible by the public that summarize the readings of all participants. Individual measurements may also be displayed on each user's mobile phone. Despite a common interest in air pollution, this application scenario and the related applications differ from the PEIR project presented in Section 2.1.1, as the pollutant concentrations are actually sensed with real sensors carried by the participants and not inferred based on weather conditions, traffic condition, and emission models.

#### *Monitoring Thermal Columns*

*Ikarus* is a participatory sensing application for paraglider pilots [vKSW11]. It collects information about thermal columns, which are used by pilots to gain in altitude during their flights. *Ikarus* is based on the collection of barometric pressure information, annotated with the time and location of sampling. Thermal maps are then extracted from the contributed sensor data, and processed for visualization on web-based maps and distribution on other pilots' navigation units.

#### *Monitoring Noise and Ambiance*

Microphones in mobile phones can be configured to measure the surrounding noise level and give insights about the nature of contextual events. In *NoiseTube* [MSNS09], *Ear-Phone* [RCK<sup>+</sup>10], *NoiseSpy* [KBRL09], and *NoiseMap* [SBS<sup>+</sup>11], noise levels are used to monitor noise pollution, which can, e.g., affect human hearing and behavior. The data are then used to build repre-

sentative pollution maps to enable specialists to understand the relationships between noise exposition and behavioral problems.

In addition to noise level, the sound samples can be further analyzed to determine, e.g., whether human voices were recorded in order to recognize the sound context in *Sound-Sense* [LPL<sup>+</sup>09]. Depending on the recognized context, the corresponding sound samples may be used to document audio diaries or indicate places where music is currently played to other users in online social networks.

Furthermore, in *MoVi* [BC10], the mobile phones collectively sense the surrounding ambiance to detect precursor signs (e.g., outburst of laughter, moves in the same directions) of relevant social events (e.g., speeches) and trigger a video recording of the upcoming events in case of positive detection. The recordings collected by different mobile phones can then be timely and automatically assembled in a common video clip, which may potentially reduce cumbersome manual video edition. As the video recordings are automatically triggered, the participants can focus on the events instead of having to focus on taking recordings.

*MetroTrack* [AML<sup>+</sup>10] is used to track mobile noise sources in outdoor environments. Similar to the aforementioned approaches, mobile phones carried by participants are used as mobile noise sensors. Collaboration between neighboring mobile phones is used to estimate the future trajectory of a noise source through the application of distributed Kalman filtering. The tracking task is automatically forwarded to nodes in proximity to the estimated trajectory to ensure accurate detection of the mobile noise source.

#### *Monitoring Road and Traffic Conditions*

The mobile phones can be exploited to document road and traffic conditions. In *Nerice* [MPR08], the embedded accelerometer, microphone, and positioning system (GPS or Global System for Mobile Communications (GSM) radio) are used to detect and localize traffic and road conditions, e.g., potholes, bumps, or braking and honking (which are both considered as implicit indicators of traffic congestion). The application integrates the provided information about the surface roughness of the roads, the surrounding noise, and the traffic conditions into traffic maps, which are available to the public. In addition to parameters related to a cyclist's activity and his physical condition, the BikeNet project (cf. Section 2.1.1) measures environmental parameters such as pollution, noise levels, and irregularities of the roads. Hence, this project can be regarded as a hybrid application, combining components from both people-centric and environment-centric sensing.

Current solutions for automotive traffic monitoring, such as inductive loops, are usually expensive in deployment and maintenance costs, and often prone to errors. The concept of *Virtual Trip Lines* [HGH<sup>+</sup>08, HIJ<sup>+</sup>12] targets to replace these conventional monitoring solutions by smartphones mounted within vehicles. Instead of deploying costly monitoring infrastructure, the locations of street segments of interest are modeled as virtual trip lines and forwarded to the participants' smartphones. A phone application constantly monitors its current location, and transmits its position and velocity to the traffic monitoring infrastructure whenever a virtual trip line has been crossed. A similar approach for traffic monitoring is based on the use of multiple positioning sensors (GPS, Wi-Fi, and cellular radios) in [TRL<sup>+</sup>09]. The *VTrack* system relies on mobile devices which deliver timestamped location estimates to a server, in order to estimate driving times on different road segments at a fine spatiotemporal granularity. Based on the availability of traffic information, roads with unusually high travel times are identified, and real-time information about these traffic hotspots is then used for route planning. An extension to the approach is presented in [TBGE10], where the concept of *cooperative transit tracking* is presented. The lack of public information about the timeliness of public transportation is encountered by a participatory approach to report the current location of transit vehicles by the participants on board. A background task on smartphones automatically detects if the user has entered a transit vehicle, both above ground and underground, and uploads its current location coordinates to the tracking server.

Although the *CarTel* [HBZ<sup>+</sup>06] and *GreenGPS* [GPA<sup>+</sup>10] projects utilize dedicated sensing devices with more resources than current mobile phones, both rely on contributions by partic-

ipants and can thus be considered as implementations of participatory sensing. GreenGPS targets to provide a map of the least fuel-consuming routes to drivers. The fuel consumption is measured via specific sensors accessing the gauges and instrumentation of the vehicles, and is correlated with location information from an external positioning system. The sensor readings are stored on a memory card and are manually uploaded to the application by the participants themselves. The obtained results have shown that high speed and long distance routes do not necessarily reduce the fuel consumption. Using an embedded computer coupled with sensors, CarTel analyzes the time it takes a user to commute to work, determines traffic congestion, and represents jammed roads on a map. Additionally, driving patterns and readings from automotive on-board diagnosis systems can be taken into account.

## 2.2 SYSTEM MODEL

From the analysis of the previous people-centric and environment-centric sensing applications, we derive the following general system model including stakeholders and architectural components. Figure 2 summarizes the resulting model and the interactions between its elements.

### 2.2.1 Stakeholders

In the above applications, we identify the following stakeholders:

- ▷ **APPLICATION ADMINISTRATORS** They are members of organization, research groups or individuals who initiate the participatory sensing campaigns. They design, implement, and deploy the application and are responsible for the maintenance and the management of the infrastructure. For example, this includes making available the application for download on the campaigns' websites or in App Stores and setting up the application server to collect and process the data.
- ▷ **PARTICIPANTS** install the sensing application on their mobile devices and voluntarily contribute to the participatory sensing campaigns by gathering sensor readings using the mobile phones they own and carry. Besides being driven by the motivation to benefit from the data provided by themselves and other participants, their contributions to the campaigns can be motivated by different factors primarily influenced by the nature of the campaign. At an individual scale, they may be willing to, e.g., improve their health conditions, monitor their impact on the environment, or document their sport experiences. In contrast at a community scale, they may aim at monitoring pollution and thermal columns to help scientists to understand the monitored phenomena, or helping other users by providing information about road and traffic conditions. Consequently, their motivations can be either self-centered (e.g., in Jog Falls), altruistic (such as in PollutionSpy), or a combination of both (e.g., in Ikarus). Note that the degree of involvement of the participants in the sensing process depends on the application characteristics, as discussed in detail in Section 2.2.2.
- ▷ **END USERS** access and consult the data gathered by the participants according to their interests and preferences. End users include, e.g., participants willing to consult their own collected data, application administrators verifying the actual contributions and results, specialized scientists attempting to gain insights about the monitored phenomena, health professionals checking patient data, or the general public.

### 2.2.2 Architecture

In Figure 2, we have identified typical architectural components common to the existing participatory sensing architectures and determined their function in the general system model.

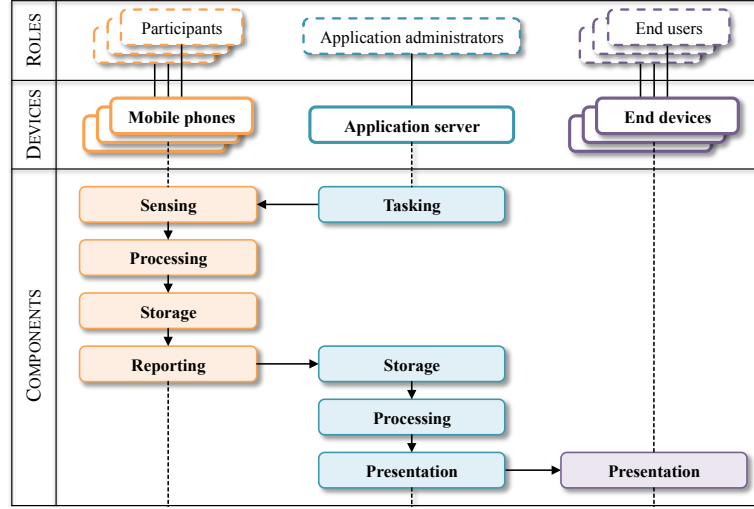


Figure 2: Stakeholders and architectural components of participatory sensing applications

The components are generally organized into a client-server architecture and interact from the sensing process to the presentation of the results to the end users.

- ▷ **SENSING COMPONENT** It is located on the participants' mobile phones and captures different kinds of sensor data, prevalently time, location, pictures, sound samples, accelerometer data, pollution data, biometric data, and barometric pressure. We summarize the sensing modalities used in the presented applications in Table 1. The sensor data can be captured according to one of the following sensing modes: Manual, automatic, and context-aware [Est10]. In the manual mode, the participants trigger the collection of sensor readings themselves when they detect relevant events, such as noise pollution or traffic congestion. This mode is also referred as *participatory sensing* in the literature [BEH<sup>+</sup>06], as the participants directly participate in the sensing process. On the contrary, the participants are not directly involved in the automatic and context-aware modes. In the automatic mode (also known as continuous mode [EML<sup>+</sup>09]), the sensor readings are collected at a constant sampling frequency, while their collection depends on the surrounding environment in the context-aware mode (also known as *opportunistic sensing* [CEL<sup>+</sup>06]), where the embedded sensors monitor their environment and activate the sensing function when previously set thresholds are exceeded.
- ▷ **TASKING COMPONENT** It supports the above sensing component by distributing the sensing tasks to the mobile phones. These tasks specify the sensing modalities based on the application requirements including criteria in order to fulfill to start the capture, the sensors to be used, and the sampling frequency. One example of such requirements could be phones equipped with GPS and with embedded cameras that can capture three megapixel images. The tasks also contain information about location and/or time frame of interest.
- ▷ **REPORTING COMPONENT** It ensures the transmission of the sensor readings collected by the sensing component to the application server. The data transfers mostly make use of communication infrastructure available to the mobile phone, such as Wi-Fi, or GSM/General Packet Radio Service (GPRS)/3G connectivity. For example, the sensor readings can be transmitted to the server using Short Message Service (SMS), Transport Control Protocol (TCP) connections [GLC<sup>+</sup>08], or remote procedure calls [MLF<sup>+</sup>08].
- ▷ **STORAGE COMPONENT** It ensures the storage of the collected data on the mobile phone and the reported data on the server. While the server manages long-term storage of the reported data, the mobile phone ensures short-term storage of the data to be processed

Table 1: Comparison of applications and sensing modalities

TYPE	MONITORED SUBJECT	APPLICATION	TIME	LOCATION	PICTURES	SOUND	ACCELERATION	POLLUTION	BIOMETRIC DATA	BAROMETRIC DATA
PEOPLE-CENTRIC SENSING	User health	DietSense [RPH <sup>+</sup> 07]	x	x	x	x				
		Pediatric obesity [AMM <sup>+</sup> 08]	x	x	x		x		x	
		BALANCE [DAC <sup>+</sup> 09]	x				x		x	x
		Jog Falls [NBB <sup>+</sup> 10]	x		x		x		x	
		HealthSense [SDAB08]	x				x			
		MobAsthma [KBRL09]	x	x				x	x	
		SenSay [SSF <sup>+</sup> 03]	x	x			x			
	Personal impact	PEIR [MRS <sup>+</sup> 09]	x	x						
	Sport experiences	BikeNet [EML <sup>+</sup> 07, EML <sup>+</sup> 09]	x	x		x	x	x	x	
		Biketastic [Shio9]	x	x		x	x			
		SkiScape [EC06, ELM <sup>+</sup> 06]	x	x		x	x			
	Social media	Micro-Blog [GLC <sup>+</sup> 08]	x	x	x	x	x		x	
		CenceMe [MLF <sup>+</sup> 08, MML <sup>+</sup> 08]	x	x	x	x	x			
ENVIRONMENT-CENTRIC SENSING	Price auditing	LiveCompare [DC09]	x	x	x					
		PetrolWatch [DKCB08]	x	x						
	Air pollution	Haze Watch [CYCS10]	x	x				x		
		PollutionSpy [KBRL09]	x	x				x		
		[PHGo7]	x	x				x		
	Thermal columns	Ikarus [vKSW11]	x	x		x				x
	Noise and ambiance	NoiseTube [MSNS09]	x	x		x				
		Ear-Phone [RCK <sup>+</sup> 10]	x	x		x				
		NoiseSpy [KBRL09]	x	x		x				
		NoiseMap [SBS <sup>+</sup> 11]	x	x		x				
		SoundSense [LPL <sup>+</sup> 09]	x	x		x				
		MoVi [BC10]	x	x		x	x			
		MetroTrack [AML <sup>+</sup> 10]	x	x		x				
	Road conditions	Nericell [MPRo8]	x	x		x	x			
		Virtual Trip Lines [HGH <sup>+</sup> 08, HIJ <sup>+</sup> 12]	x	x						
		VTrack [TRL <sup>+</sup> 09]	x	x						
		Transit tracking [TBGE10]	x	x			x			
		CarTel [HBZ <sup>+</sup> 06]	x	x	x		x	x		
		GreenGPS [GPA <sup>+</sup> 10]	x	x						

or transmitted to the server. On the server side, the data are commonly stored in relational databases [GLC<sup>+</sup>08, HBZ<sup>+</sup>06, KGZ07] or databases specially adapted to the management of sensor readings, e.g., sensedDB [GKN<sup>+</sup>07] and SensorBase [CYHE06].

- ▷ **PROCESSING COMPONENT** Depending on the application, the function of the processing component is distributed between the mobile phones and the application server. For example, features of interest, such as the noise level, can be directly extracted from the sound samples on the mobile phones on an individual basis. On the server side, the reported sensor readings are processed at larger scale to, e.g., compute statistics and



build summaries over several users. It hence prepares the data for the presentation component.

- ▷ **PRESENTATION COMPONENT** It presents the results obtained by the processing components to the end users. Individual results stored and processed locally can be directly display on the mobile phones of the respective participants. Alternatively, consolidated results of multiple users can be presented through web-portals to a larger public, depending on the characteristics of the applications. The results are presented in forms of raw data to allow end users to analyze them themselves, or in forms of graphs, maps, and geographic overlays [HBZ<sup>+</sup>06]. End users can also send queries to the application server in order to access sensor readings collected by other users.

## 2.3 PRIVACY

As we have illustrated in Table 1, virtually all participatory sensing systems collect sensor readings related to the participants and/or their environments. Obviously, the collected data may be used to extract or infer sensitive information about a user's "private life, habits, act and relations" – the basic definition of *privacy* by Brandeis [BW90]. Simultaneously, contributed sensor data are vital to any participatory sensing application, and their deficiency endangers the success of participatory sensing systems. Application administrators therefore need to increase the user awareness of the consequences of the disclosure of sensor data as well as provide solutions to maintain user privacy in order to ensure the durability of the campaign and prevent participants from opting out.

In this section, we first present our definition of privacy, specially tailored to participatory sensing systems, and used throughout the remainder of this thesis. We then conduct a privacy analysis and highlight actors and processes that represent threats to privacy. We finally outline possible consequences resulting from the disclosure of privacy-sensitive information.

### 2.3.1 Definition

With the rise of communication equipment and computing systems, the notion of *information privacy* has emerged, which has been initially defined as "the claim of individuals [...] to determine for themselves when, how and to what extent information about them is communicated to others" by Westin [Wes67]. Although the field of privacy is multifaceted and comprises several other dimensions [RGC10], user-level control over sensitive sensor data represents the major concern in participatory sensing systems. In consequence, we refer to information privacy whenever the term "privacy" is being used in this thesis.

In order to cater for the specific characteristics of participatory sensing systems, we propose an adapted version of the above definition of information privacy, which we apply in the remainder of this thesis:

Privacy in participatory sensing is the guarantee that participants maintain control over the release of their sensitive information. This includes the protection of information that can be inferred from both the sensor readings themselves as well as from the interaction of the participants with the participatory sensing system.

Our definition implies that participants continuously need to control the release of their sensor readings to third parties (including the participatory sensing application) and have full control over the provided type of sensor readings, the degree of granularity, the spatiotemporal context, and the data recipients. In addition to the direct protection of the sensor readings, the proposed definition includes the protection of the participants against the inference of sensitive information resulting from their interactions with the participatory sensing system. This implies that potential adversaries are unable to determine the spatiotemporal information about the participants, when they download tasks or report data to the application, or link their real identity with the pseudonyms they use.

### 2.3.2 Privacy Analysis

To further understand the privacy concerns in participatory sensing, we analyze participatory systems from a privacy perspective. We base our analysis on the theory of *contextual integrity* [Niso4], which comprises the dimensions of *appropriateness* and *distribution*. Appropriateness defines if the revelation of a particular piece of information is appropriate in a given context, while distribution focuses on the occurrence of an information transfer from one party to another. The concept of *contextual integrity* defines breaches of appropriateness or distribution as violations to a user's privacy.

Socio-cultural and contextual differences have a strong impact on the individual perception of data sensitivity. For example, participants make different privacy decisions depending on the number of recipients of their data [TLH<sup>+</sup>10]. While it has been shown that users pragmatically determine whether they share their locations with a single person, additional parameters like their willingness to attract attention or boost their self-presentation are taken into account when they decide about sharing their whereabouts with a larger group of people. In consequence, the appropriateness entirely depends on the individual privacy conception of each user. We thus discuss this dimension at a high level and confine our privacy analysis to a thorough examination of distribution aspects for the stakeholders present in typical participatory sensing applications. In all cases, we assume that the analyzed participatory sensing systems collect sensor readings including pictures, sound samples, acceleration, pollution data, biometric data, and barometric pressure (see Table 1), which are tagged with spatiotemporal information.

- ▷ **WHO GATHERS THE SENSOR READINGS?** The sensor readings initially collected by the participants are reported to the application server, which is run by the application administrators. The administrators therefore have a direct access and control over the collected sensor readings. While the distribution of the sensor readings to the application administrators is part of participatory sensing systems and the participants are aware of this fact, the reported sensor readings may reveal sensitive information about the participants, if no privacy-preserving processing is locally applied on the mobile phones. The participants are the only ones able to judge the appropriateness of the revealed information, as it depends on their personal privacy conception as well as the nature and the context of the revealed information. However, the users may encounter difficulty in translating their own conception of privacy into, e.g., privacy settings, and to understand the implications of their settings and actions on their privacy [DLHH09]. For example, participants are often unaware of the technical details of the underlying architecture. They may underestimate the risks related to their privacy [AG05], and pay little attention to policies and end user licensing agreement, both of which are often laid out in technical terms and thus hard to understand [GDG<sup>+</sup>05]. In summary, a combination of the inconsistent behavior of the users and the lack of existing solutions to clarify privacy implications may lead to sub-optimal privacy protection.
- ▷ **WHO ANALYZES THE SENSOR READINGS?** The sensor readings are analyzed after having been reported to the servers. The application administrators determine and implement the processing applied on the sensor readings. Particular privacy-aware functions can be applied to anonymize the users and/or remove sensitive information about them before their release to the analysts. However, the sensor readings can still contain private information about the users. After this preparatory processing, the sensor readings can be analyzed by different groups of people depending on the application scenario. These include the users themselves, the application administrators, doctors, researchers in the field, etc. The users maintain different relationships with the groups of analysts that directly impact the appropriateness. For example, it is considered as appropriate for users to share personal information related to their health conditions with their doctors [Niso4], whereas sharing the same information with a different analyst, such as a application administrator, may be inappropriate. By reporting



their sensor readings to the application, the users are aware that they can be analyzed by the application (i.e., the application administrators). However, they may ignore that the distribution of their sensor readings can be extended to external analysts potentially unknown to them, endangering the appropriateness.

- ▷ **WHO ACCESSES THE PROCESSED SENSOR READINGS?** The analyzed sensor readings are released by the application administrators to the end users. The access to the sensor readings is determined by access control rules defined by either the users themselves or the campaign type. The authorized persons can be restricted to only the users themselves or extended to their relatives, friends, or a larger public, depending on the application. In the worst case, the circle of authorized persons is thus enlarged from the application administrators and analysts to the general public, raising the issue of appropriateness to its climax. The level of appropriateness can be moderated by proposing different degrees of granularity at which the data are released depending on the nature of the relationships between the users and the end users. However, the distribution of the sensor readings is performed by the application administrators. The users must thus trust them not to disclose sensitive information about themselves to untrusted parties.

In summary, the respect of the privacy of the users therefore primarily depends on the application administrators who have direct access to the reported sensor readings and ensure their distribution to potential analysts and end users. Malicious application administrators or inefficient mechanisms to both remove private information from the sensor readings and control their access can thus contribute to the violation of the privacy of the users.

### 2.3.3 Privacy Threats

Let us assume that a worst case scenario where application administrators break the trusted relationship to the users and reveal sensitive information about them. We examine the potential social consequences of such disclosure by successively considering the sensing modalities listed in Table 1:

- ▷ **TIME AND LOCATION** It is evident by examining the table that virtually all applications (except for Jog Falls, HealthSense, and BALANCE) collect time and location information independently of their people-centric or environmental-centric nature, thus underpinning the importance of these two contextual factors. GPS receivers embedded in most current smartphones provide very accurate location coordinates. However, in the absence of GPS (due to lack of coverage or if the user does not want to reveal fine-grained location information), Wi-Fi, or cellular network based triangulation can be used to obtain coarse-grained location information [LCC<sup>+</sup>05]. Contextual information collected from other embedded sensors (such as points of interest [ACC09], light, and noise) can also be used to identify a person's location. Furthermore, temporal annotations of sensor data can provide insights about the habits of the users.

Given their importance, the disclosure of data from these two modalities has been shown to leak privacy-sensitive information about the users, including their home and workplace locations, as well as their routines and habits [Shio9]. For example, frequent visits to hospitals may allow employers to infer the medical condition of their employees, and similarly, attendance at political events may provide information about the political views of users [Liu07]. In summary, without any protection mechanism, the disclosure of location information may lead to severe consequences ranging from social to safety and security threats [Shio9]. Additionally, the threats resulting from location/time traces are not confined to applications where authentication is required. Even in the case of anonymous contributions, location traces may be analyzed to infer the identity of the users based on their residence location and reverse white page lookups [Kru07].

- ▷ **SOUND SAMPLES** Besides inferring identities and preferences from spatiotemporal data only, the portrait of the user can be refined by complementing these data by samples of

other sensing modalities. In several of the aforementioned applications, sound samples are either recorded intentionally by the users, or captured automatically by the mobile phones. While users can easily preserve their privacy by only recording non-sensitive events in the former case, mobile phones effectively behave as smart spies in the case of automated recordings. Dedicated user interaction is required to prevent the applications from recording private conversations about intimate or confidential subjects. Even in public locations, the recognition of characteristic sound patterns that are unique to certain events and locations may allow adversaries to determine a user's current context.

- ▷ **PICTURES AND VIDEOS** The content of contributed pictures and recorded videos is also likely to reveal personal information about the users and their environment. Although DietSense [RPH<sup>+</sup>07] targets to take photos of consumed meals, no countermeasures are taken to conceal the faces of persons sharing their meal with the users. In all scenarios, in which the camera is oriented away from the user, faces of other people in the vicinity are possibly captured in the images, and thus conclusions about the number and identity of the user's social relations can be drawn. The publication of captured pictures may lead to similar consequences as in online social networks, such as Facebook, where a teacher was suspended due to a picture showing her holding glasses filled with alcohol [CBS11], or a depressed woman who lost benefits from her health insurance for pictures showing her attending parties and relaxing on the beach [CBC09]. Similar to sound recordings, the current user context and the surrounding environment may also be extracted from sensor data. For example, pictures showing points of interest may easily establish the user's presence at those locations.
- ▷ **ACCELERATION** Raw accelerometer readings may appear less threatening in revealing private information about the users. However, this hypothesis is not always true and may often only serve as a false sense of security. For example, if the mobile phone is carried on the hip, information about the gait, and thus possible indications about a user's identity, may be inferred [DNBB10]. Additionally, the research field of activity recognition also makes extensive use of accelerometer readings [GFH09]. The exploitation of these data by malicious users may have negative consequences. For example, employers may want to verify that their employees are actually working during their working hours. If the employers detect anomalies, they might suspend the respective employees. Moreover, acceleration data may be exploited to infer the current activity of the users or text sequences they entered on their mobile phone [OHD<sup>+</sup>12].
- ▷ **ENVIRONMENTAL DATA** Recording particles and gas concentrations or barometric pressure may not directly threaten the privacy of the users by themselves. However, particular air compositions combined with secondary information, such as precise air temperature, might identify the location of the users at a level of granularity as fine as room levels within buildings, where location information can be inaccurate due to non-availability of GPS or other location services.
- ▷ **BIOMETRIC DATA** Biometric sensor data can be used for a diagnosis of a user's current physiological state. Similarly to medical staff, adversaries may identify health anomalies or diseases based on the captured sensor data. Leaked medical information may then be used by health insurance companies or employers to revoke contracts, if an impairment of the physiological conditions of the users is identified. Biometric data can also be used in authentication mechanisms. Their leakage therefore poses the risks that unauthorized persons access sensitive data or locations.

Privacy threats represent an inherent problem of any participatory sensing application. Although the subjects of interests of environment-centric applications are not the users themselves, all considered applications monitor the spatiotemporal context of the users and therefore represent a danger to their privacy. Furthermore, additionally captured sensing modalities may provide further insights about the users. As a result, environment-centric

applications can similarly endanger the privacy of the users, even if the threats are less perceptible at first sight than in the case of people-centric applications. Further sensor modalities can be anticipated in future, such as using stylus or touch screens or wireless interfaces (e.g., Near Field Communication (NFC), Wi-Fi, or Bluetooth) for capturing personal information about the users and other users in their surroundings.

Location privacy has been predominately addressed in the literature in comparison with the other sensing modalities. The particular interest for location privacy may be due to the sharing of similar concerns with orthogonal domains addressing this issue, such as vehicular networks [RH07], location-based services [MFD03], pervasive computing [BS03], ubiquitous computing [Bero5], etc. In contrast, the remaining sensing modalities are less represented in the literature, as they are one of the particularities of participatory sensing applications. However, we have shown in this section that their combinations may leak sensitive information about the users and people in their vicinity, or provide information about their locations, even if those are protected by privacy-preserving mechanisms.

## 2.4 SUMMARY

Participatory sensing leverages the ubiquity of mobile phones to open new perspectives in terms of sensing. Within the scope of this chapter, we have analyzed existing participatory sensing applications. As a result, we have identified the different sensor modalities contributed by the participants of such applications and mapped the components of these applications into a generic system model. Our analysis has revealed that virtually all applications capture location and time information. This information is used either as self-contained data, or to geo-tag and timestamp other collected sensor readings including pictures, sound samples, acceleration, pollution, and biometric data. We have examined the extent of personal information that can be inferred from the collected sensing modalities, both individually and in combination. Finally, we have provided a definition of privacy in participatory sensing and highlighted potential threats to user privacy resulting from the uncontrolled disclosure of personal information to untrusted people.



After having outlined the privacy threats and the corresponding need for privacy-protecting mechanisms to encourage user participation, we examine the current state-of-the-art solutions which attempt to address these threats. We conduct a cross-analysis of the architectural elements present in typical sensing applications and present existing countermeasures based on our work [CRKH11]. In particular, we focus on solutions that have been applied in the specific domain of participatory sensing. Figure 3 depicts the several stages through which the sensed data pass between its collection and the consumption by the target audience and summarizes potential countermeasures to privacy threats applied at each step. By tracing the distribution path of sensor data, we outline how privacy can be maintained and improved at these steps. Firstly, we address the implications on privacy as outlined in Chapter 2 by discussing tailored sensing and anonymous task distribution in Sections 3.1 and 3.2, respectively. We then consider different schemes to anonymize and protect the privacy of the users while the data are reported to the application server (see Section 3.3). Subsequently, we detail solutions for privacy-aware data processing in Section 3.4 and storage including mechanisms to review, delete, and control the retention of sensed data in Section 3.5. We next present current solutions to control and audit their access in Section 3.6 and conclude the discussion of countermeasures by focusing on data queries in Section 3.7. Note that we subsequently discuss related work specific to our contributions in each corresponding chapter.

### 3.1 TAILORED SENSING AND USER PREFERENCES

A first measure to encounter the privacy threats discussed in Section 2.3.3 is to control the data collection process at the user level and allow the users to express their privacy preferences. In the presented scenarios, this control is applied to different extents. Although some solutions allow users to fully disable the sensing function [MLF<sup>+</sup>08, SBE<sup>+</sup>08], doing so is of little use for participatory sensing, since users would be unable to contribute any data. As proposed in [DMP<sup>+</sup>10], this binary scheme (full access to sensor data, or none at all) can be extended by introducing additional intermediate levels. For example, the users may decide to selectively enable sensor measurements depending on a variety of factors, such as presence in sensitive locations (home or office), or their current social surroundings (presence of friends or family members). The selection of these factors allows users to explicitly indicate the type of information that they are happy with being collected in different contexts. They thus define their personal conception of appropriateness as defined in Section 2.3.2.

To reflect the user's privacy preferences whilst optimizing data fidelity, a finer-grained scheme allows users to adjust the sensing granularity and the time resolution. For example, samples may be collected every hour instead of every 15 seconds, or location information be captured at different degrees of granularity [SLB<sup>+</sup>03]. Table 2 illustrates possible degrees of granularity and related examples which may be applied in the context of some of the applications discussed in Section 2.1. Starting with the finest granularity on top, i.e., the unaltered original/raw data, the data resolution decreases towards the bottom of the table until reaching the coarsest meaningful degree of granularity. Depending on the application characteristics, the granularity of the different sensing modalities can be tuned in different ways, with the ones presented in the table only serving as illustrative examples.

Although this scheme does not provide a generic trade-off between privacy and disclosed data, it is expected to improve the overall acceptance of the solution by offering additional

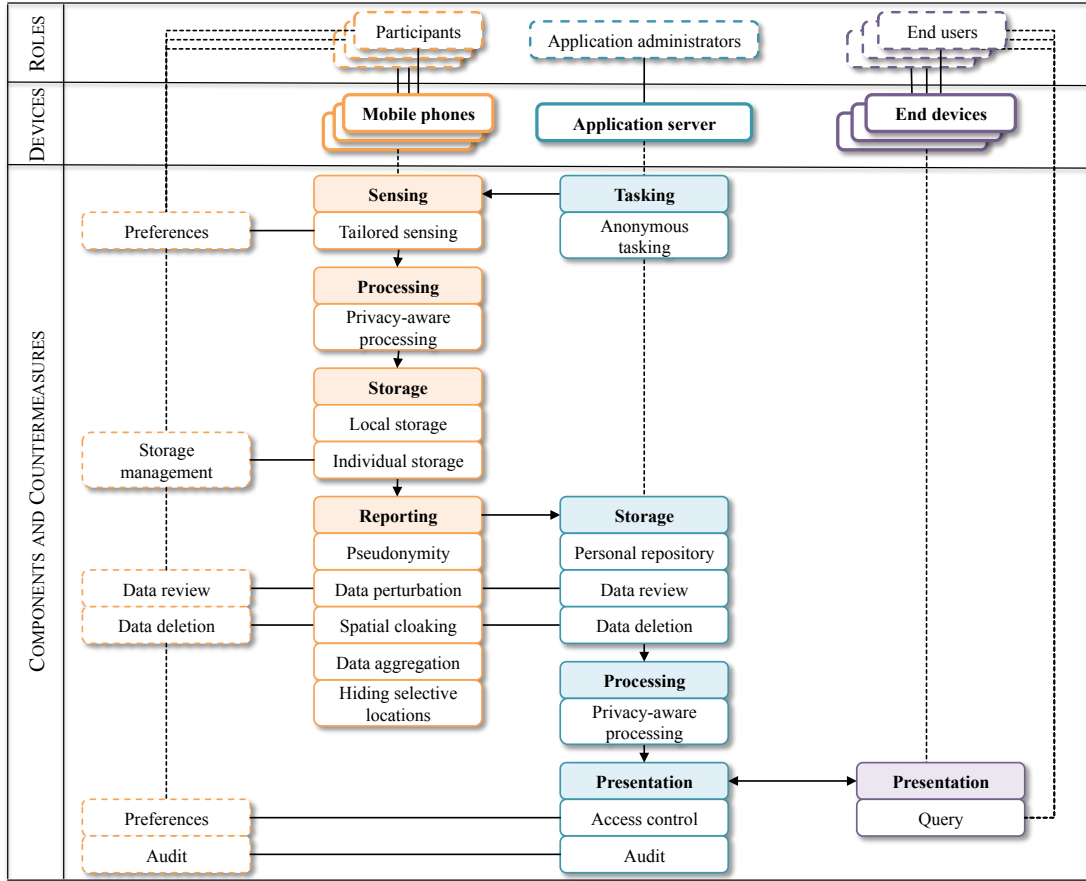


Figure 3: Countermeasures and their relationships to the architectural components

Table 2: Example of granularity degrees for different sensing modalities

GRANULARITY	LOCATION	SOUND	PHOTO	ACCELERATION
Fine-grained	Precise position	Original sample	Original image	Raw data
....	Street	Voices removed	Faces blurred	Activity type
..	District	Spectral properties	Number of persons	Activity category
Coarse	City	Loudness level	Environment (indoor/outdoor)	Motion (yes/no)

resolutions of granularity to the users. From the perspective of the application, coarse-grained data are still better than no data. For example, the location of air pollution measurements can be released at the granularity of a district. It should also be noted that sensing granularity has a direct impact on the energy consumption of the mobile phone. Whenever a sensor is activated for collecting data, it consumes energy from the phone's battery. As such, sampling at a coarser granularity implies that the sensors may be turned off for longer periods of time, thus resulting in energy savings. Simultaneously, both the size and quantity of the data to be transmitted are reduced. Transmitting data also consumes energy for the radio (3G or Wi-Fi) transmission, thus less data volume also results in additional energy savings. However, additional processing may be required to filter the data and report them with a coarser granularity (e.g., recognizing the presence of faces on the images to eliminate them) leading to supplementary energy consumption, which needs to be regarded specifically.

### 3.2 PRIVACY-PRESERVING TASK DISTRIBUTION

Sensor data collection is generally triggered through *tasks*, which specify the sensing modalities (e.g., regions of interest, criteria to fulfill to start the capture, sensors used, and sampling frequency) based on the application's requirements. The tasks are subsequently distributed to the mobile phones that satisfy the tasking requirements. The distribution of tasks can, however, threaten the privacy of the participants. In this section, we first outline the different existing modalities to distribute sensing tasks, before highlighting specific threats to privacy and summarizing existing solutions.

The tasks are initially downloaded from the application server. A central *tasking component* located on the application server or a dedicated *tasking server* [KGZ07] selects appropriate devices based on predetermined criteria to optimize the sensing process, such as the current location of the mobile phone or its available resources (embedded sensors, battery lifetime, currently executed tasks, or processing capabilities) [RSB<sup>+</sup>09]. Once the devices are selected in a centralized task distribution model, the application server deploys the sensing tasks on the devices, e.g., via a push model based on executable binaries [DMP<sup>+</sup>10].

In case of decentralized task distribution, mobile phones autonomously manage and distribute tasks to devices in their proximity, depending on the availability of the required sensors, such as proposed in [ELCo8]. For example, a mobile phone A, which is not equipped with the sensors required by the tasks, transfers its tasks to other mobile phones embedding such sensors in the vicinity. A initiates the task transfer by broadcasting requests including the task to execute and the corresponding sensors to mobile phones B and C in its proximity. At the reception of these requests, B and C verify if they can fulfill the included task conditions. Assuming that only B possesses the required sensors, A transfers its sensing task to B. B can fulfill the transferred task either instantly if both mobile phones are located in the task's region of interest or remotely when B enters the region of interest.

The Bubble-sensing model [LLECo8, LLECo10] is a hybrid alternative to the purely centralized and decentralized schemes. Although mainly based on decentralized distribution of tasks, the concept also requires the presence of a central entity to maintain the sensing tasks. Users, designated as *bubble creators*, can create persistent sensing areas at defined places of interests. They initiate a sensing request including the geographical region, duration, and sensing modality corresponding to each bubble and broadcast it to potential *bubble carriers* (i.e., other users) in a distributed fashion. Bubble carriers may then move to the specified location, perform the sensing, and report the collected data to the central entity, the *bubble server*, from where the data can be retrieved by the bubble creator. The persistence of bubbles during the sensing period is ensured by designating bubble carriers with low expected node mobility as *bubble anchors*, which maintain bubbles on behalf of their creators, should they become disconnected from the bubble. Nevertheless, bubbles may disappear in absence of anchors. In order to address this issue and restore orphaned bubbles, the users additionally contact the bubble server in regular time intervals to search for bubbles near their current location, and join them if the required sensing modalities match.

In all three presented approaches for task distribution, the tasking and downloading processes may endanger the privacy of the users in several ways. First, downloading tasks provides information to the tasking server about the location of the users at precise timestamps, while the nature of the tasks provides hints about the devices used. Additionally, even when pseudonyms are used, the tasking server may infer the identity of the users by tracking their locations across multiple downloads, as the locations of their workplaces and homes may be exposed [Kru07]. In order to protect the users, the following mechanisms for ensuring anonymity and location-privacy have been proposed:

- ▷ **TASKING BEACONS** The participating devices receive the broadcast beacons including the sensing tasks without having to register/authenticate themselves to a central entity [KKT09]. As a result, users do not expose their identity to the server through the broadcast nature of the beacons.



- ▷ **TASKS DOWNLOADS IN DENSELY POPULATED LOCATIONS** The high density of people present at such locations makes the identification of the users by the server difficult, and hence conceals their identities [SCP<sup>+</sup>10]. As a downside, users can only download tasks in highly populated areas.
- ▷ **ATTRIBUTE-BASED AUTHENTICATION** Instead of using their precise identity, the users can use cryptographic-based credentials showing their memberships to a particular group (e.g., students registered in the cycling club of the university) [KKT09], as realized in [CS97]. Again, the identities are hidden within the group of users and the level of protection depends on the size of this group.
- ▷ **LOCATION PRIVACY-PRESERVING ROUTING SCHEMES** Although not directly anonymizing the users, these schemes hide their location using specific router/relay organization [AMCK<sup>+</sup>02, DMS04]. For example, the TOR-based routing scheme used in [SCP<sup>+</sup>10] anonymizes the connections to the tasking server using multiple relays and onion routing to hide the Internet Protocol (IP) address, and thus information about the current location of the users [KKT09].

Moreover, a malicious tasking entity may submit tasks with restrictive acceptance conditions including, e.g., a rare sensor type or specific mobile phone brands. Known as *narrow tasking* [SCP<sup>+</sup>10], this attack may allow the attacker to de-anonymize the mobile phones accepting these highly device-specific tasks, as only one or few mobile phones share these restrictive conditions. A countermeasure to the narrow tasking attack consists of introducing a trusted third party storing the attributes of all mobile phones and verifying that a sufficient number of mobile phones (above a predetermined threshold) are able to fulfill the acceptance conditions in order to protect the anonymity of the mobile phones and their users accepting the tasks.

Furthermore, a malicious tasking entity may attempt to differentiate and identify anonymous users by launching *selective tasking* attacks [SCP<sup>+</sup>10], where the tasking entity distributes a task to only a restricted pool of mobile phones. The selective tasking attack differs from the narrow tasking attack, as the selective tasking attack aims at linking the anonymous mobile phones uploading the tasking reports to the reports themselves, while the narrow tasking attack straightforwardly infers the identity of the devices/users based on the task acceptance. As the number of tasked devices is restricted, the adversary can easily link each anonymous mobile phone to the reports it has uploaded. A further analysis of the uploaded reports may then breach the anonymity of mobile phones. In order to prevent this attack, the responsibility of selecting the tasks can be transferred from the tasking server to the users themselves, who select a random amount of available tasks to execute.

### 3.3 ANONYMOUS AND PRIVACY-PRESERVING DATA REPORTING

In most applications, captured sensor data are reported to the central server directly after they have been recorded. We have identified that almost exclusively, all participatory sensing applications record location and time information (cf. Section 2.3.3). A prominent attack is thus the inference of location traces. The collection of location information over several reports may allow to identify the paths followed and the locations visited by the participants. The disclosure of the raw location data is likely to reveal the identity of users and may thus endanger their privacy. Besides the actual data contained in the reports, reporting data to the application server may also threaten the privacy of the users. In fact, users' locations can be inferred from the IP address associated with each upload using IP address geolocation. By analyzing the reporting pattern across multiple reports, additional conclusions about the whereabouts of the users can be drawn that may lead to their identification and de-anonymization [SCP<sup>+</sup>10]. In this section, we summarize existing mechanisms used to protect the privacy of the users against the analysis of both report contents and reporting patterns.



### 3.3.1 Pseudonymity

A common mechanism to protect the anonymity and privacy of the users is the use of pseudonyms (cf. Section 3.2). Instead of transmitting names in plain text, all interaction with the application is performed under an alias. Pseudonymity is currently used in various applications, including [SBE<sup>+</sup>08, Shio9, DCo9]. When used in conjunction with authentication mechanisms, pseudonym-based solutions suggest anonymity and confidentiality to the user [SBE<sup>+</sup>08]. The users tend to share their sensor readings without apprehension as they feel more protected behind pseudonyms. This subjective feeling however leads to a false perception of security, as the use of pseudonyms does not necessarily guarantee privacy in location-based applications. As demonstrated in [Kru07], an analysis of the reported data in conjunction with the reporting patterns may allow identifying the users' residences among other significant places, such as workplaces and favorite entertainment centers, based on their location traces. The residence addresses may then be exploited to find the corresponding users' real names using reverse address lookups. Pseudonyms must therefore be complemented by additional mechanisms to protect the users' locations (during both the sensing and reporting processes) to efficiently provide privacy guarantees. The anonymity-based TOR network [DMS04] is used in [SCP<sup>+</sup>10] to hide the origins of reports and prevent an identification of the real identities of the users from their locations. Before transmitting reports, the mobile phones select random relays along the path to the application server, instead of a direct route. The selected routes are then appended to the reports using a layered scheme, similar to onion layers. At each relay on the selected routes, a layer is removed using a symmetric key shared between this relay and the mobile phone. As a result, no relay knows the complete path from the report's source to the application server, but only the identities of preceding and following hops/relays.

### 3.3.2 Spatial Cloaking

In addition to privacy-aware routing (e.g., TOR-based networks), mechanisms based on  $k$ -anonymity [Swe02] can be applied to protect the location privacy of the users who upload reports. The key idea behind  $k$ -anonymity is to build groups of  $k$  users or reports such that they share a common attribute (e.g.,  $k$  users located in the same district), rendering them indistinguishable from each other. Different methods can be used to find an appropriate and common attribute in order to construct groups of  $k$  users. These methods can be classified into the two main categories of *generalization* and *perturbation* [HKH10b].

In the former, the original value of the attribute is generalized by a value with less degree of detail. For example, the exact coordinates of the  $k$  users are replaced by the name of the district of their current location. In contrast, perturbation is based on replacing the original sensor data by new values resulting from a function applied to the  $k$  sensor readings of the group members. For example, the location of each group member can be replaced by the average location of all members.

Tessellation is a form of generalization based on the division of a geographic area into multiple tiles. It is applied in [SCP<sup>+</sup>10], where tiles are established by applying Voronoi partitioning to a map of Wireless LAN access points. The access point in the center of each tile keeps track of the average number of connected devices, which is equal to the maximum value of  $k_t$  that can be achieved within the tile. To guarantee  $k$ -anonymity throughout the network, neighboring tiles with  $k_t < k$  are combined into cells with an effective value equal to the sum of each individual tile. Once the cells have been defined, the users tag their sensor data with the geographical boundary of their current cell instead of supplying their exact coordinates. Alternatively, instead of transmitting the dimensions of the cell, the users may report the geographical center of the cell, as proposed in [HKH10b].

In contrast, microaggregation [DFMS02] does not generalize the user location to a cell, but replaces the real location by the averaged location of the  $k$  nearest users, the so called *equivalence class*. Setting up equivalence classes is known to be NP-hard [Bru78], and among the proposed heuristics, the Maximum Distance to Average Vector (MDAV) algorithm has

been shown to be very efficient in setting up equivalence classes. However, as user mobility is inherent in participatory sensing applications, equivalence classes may need to be changed dynamically and thus protect new users less efficiently. Frequent re-computations in turn may negate the efficiency advantage of MDAV. The Variable-size Maximum Distance to Average Vector (V-MDAV) [SMBDFo6] algorithm offers variable class sizes and improves the performance of its predecessor in dynamic environments. The recursive algorithm is composed of two main steps. In the first step,  $k$  users are clustered based on their locations and the relative Euclidean distances. Once the clusters are formed, additional users can join the clusters in the second step, even if each cluster already contains  $k$  members and thereby fulfills the  $k$ -anonymity requirement.

A risk to  $k$ -anonymity is the possibility of homogeneity attacks, as outlined in [MKGV07]. Such attacks exploit the monotony of certain attributes to identify individuals from the set of  $k$  users. The authors thus present an extension to  $k$ -anonymity, termed  $l$ -diversity, which additionally requires the group members to provide at least  $l$  different values for the sensed attribute of interest. As a result, at least  $l$  distinct values for the sensitive attributes are present within each user group, which represents an effective countermeasure to homogeneity attacks. The principle of  $l$ -diversity is employed in [HKH10b], where an  $l$ -diverse extension to the aforementioned V-MDAV algorithm is proposed and evaluated. The authors also extend the V-MDAV scheme further by introducing Hybrid V-MDAV, which combines V-MDAV with tessellation (with tile center reporting). The hybrid scheme thus benefits from the advantages of both approaches. Tessellation is applied if a user can construct a tile within his own cell, meaning that at least  $k$  other users share the same cell. Otherwise, the V-MDAV scheme is used, as it performs better in case of sparse distribution of users across multiple cells.

Nevertheless, these approaches rely on a trusted third-party managing the generalization or perturbation of the locations for all users. To generate the cloaked values, the users need to report their exact locations to the third-party entity. In contrast, [APN<sup>+</sup>12a, APN<sup>+</sup>12b] propose an approach, in which the mobile phones locally and dynamically evaluate the degree of spatial cloaking to apply. Based on the past and current mobility patterns of the users, the clients iteratively reduce the degree of granularity of the released locations until reaching a privacy threshold set by the users.

### 3.3.3 Data Perturbation

Data perturbation intentionally perturbs the sensor samples by adding artificial distortions such as Gaussian noise to the data on the client side. The overall intention of data perturbation is to determine community trends and distributions without revealing individual data. The characteristics of the applied noise must thus be chosen carefully, as it needs to perturb individual sensor readings sufficiently while still ensuring that the statistical trend remains unaffected. For example, independent random noise has been demonstrated to be insufficient to prevent adversaries from reconstructing the original data in [Kru07].

A data perturbation scheme particularly tailored to the requirements of participatory sensing applications was proposed in [GPTA08]. Its principle is as follows. First, a noise model with characteristics similar to a realistic data set is generated using an approximate model of the phenomenon monitored by the application. Note that preliminary knowledge about the data distribution is required, which may not be available in all participatory sensing applications. The designed model, which is composed of a structural description as well as the probability distribution of the parameters, is then distributed to the community. The users use the distributed model to locally generate noise and superimpose it on their sensor readings. In order to complicate the reconstruction of each individual data, the proposed approach allows the users to change the values of the noise generation parameters regularly. The data perturbed by the users are then reported to the application. As the statistical characteristics of the noise model are known, the sum, average, and distribution of the added noise over the data of all users can be approximated. The community results (including trends and distribution) can be estimated by subtracting the average noise time series from the sum of all

individual perturbed data, the precision of the estimation increasing with the community size. Other examples that make use of data perturbation, though not in the context of participatory sensing, can be found in [AS00, AA01, EGS03].

#### 3.3.4 Hiding Sensitive Locations

Sensitive locations can be selected by the users and protected using location selective hiding [MRS<sup>+</sup>09], which was introduced in the PEIR campaign presented in Section 2.1.1 and represents an alternative to data perturbation. When the user approaches a location which has been priorly defined as sensitive, the application generates fictitious location traces which intentionally avoid the selected location. However, the generated traces remain realistic, i.e., following exiting roads and streets. The algorithm selects the closest routes first, then refines the selection by taking the history of the users' results into account (e.g., their physical capacities based on their preceding experiences). Furthermore, the algorithm even shifts the activities timely and modifies their duration to maintain the consistency of the application results. In comparison with data perturbation (Section 3.3.3) and spatial cloaking (Section 3.3.2), the location selective hiding scheme improves the location privacy without impacting the application results. In fact, data perturbation and spatial cloaking only modify the location information without considering the consistency of the results and visits to the sensitive locations may still be identifiable after the application of these schemes depending on the granularity and noise model selected.

#### 3.3.5 Data Aggregation

In comparison to the previous schemes, the privacy-preserving data aggregation approach proposed in [SZLZ10] does not rely on a central entity to protect data privacy, but on a mutual protection between users. Before transmitting data to the server, the mobile phones partially distribute their data among their neighbors. The mobile phones then upload the sensed data coming from their neighbors and the remaining of their own data. This distribution diminishes the probability to successfully attribute each sensor reading to the mobile phone which actually captured it. For example, if two mobile phones A and B exchange half of their data, the probability that the data reported by A was actually captured by itself is 50% (in absence of any additional and prior knowledge), and the same holds for B.

Moreover, this approach does not require any preliminary knowledge about the data distribution which is necessary in data perturbation schemes (cf. Section 3.3.3). Depending on the nature of the aggregation functions, two distinct schemes can be applied. For additive functions, each mobile phone/node partitions its data into  $n + 1$  slices and sends one slice to each of  $n$  selected nodes. There are three ways in which nodes can be selected. In the first model, the nodes are selected randomly regardless of their location leading to an additional energy consumption overhead if multi-hop communication is supported. In the second model, the one-hop neighbors are selected via a single broadcast. The efficiency of the privacy protection directly depends on the density of neighbors located in the broadcast regions, as  $n$  is lower in sparsely populated regions than in dense ones. An  $h$ -hop version is proposed in the third model, where  $h$  is a system parameter. Once each node has distributed its slices to its neighbors, the exchanged slices and the node's own slice are combined and sent to the aggregation server which is then able to compute the aggregation result. For non-additive aggregation functions, such as percentiles and histograms, a method combining slicing, count query, and binary search can be applied. Nevertheless, this approach only ensures data privacy protection if the nodes and the server do not conspire to breach the privacy of potential targets.

### 3.4 PRIVACY-AWARE DATA PROCESSING

In typical participatory sensing applications, data processing is shared between the mobile phones and the application server. However, due to the resource constraints on mobile platforms, the distribution of the processing tasks between both parties is typically biased towards the server. While preprocessing is generally carried out on the phones to reduce the amount of data to transfer in order to save bandwidth and transmission energy, complex processing tasks may exceed the computational power of mobile phones, making their execution on the server necessary. Moreover, complex processing might also require information from other participants, which complicates its execution on mobile phones.

In the applications discussed in Section 2.1, the data processing on the mobile phone mainly constitutes extracting features from the raw data to remove sensitive information endangering the privacy of the users (e.g., human voices recorded or people photographed) and for resource saving purposes. For example, an audio classifier can analyze the sound samples to determine whether human voices were recorded [MLF<sup>+</sup>08]. Further, the loudness level of the audio samples can be determined locally by running signal processing algorithms to minimize the data to be transferred to the server [MSNS09]. After processing, the raw data may be deleted from the local storage and the processed summaries are reported to the central server.

On the server side, the reported data may then be processed to eliminate privacy-sensitive information, such as the identity or data characteristics threatening the anonymity/privacy of the users. For example, the captured data can be aggregated among several users to render them indistinguishable or published in the form of statistics [GPA<sup>+</sup>10] and maps [DKCB08]. By doing so, sensitive data are not directly revealed to the end users, which avoids direct identification of the users. However, the users must rely on the application to efficiently anonymize the data, sufficiently protect their privacy, and not disclose the privacy-sensitive information contained in their reported data to third parties.

### 3.5 REVIEW, DELETION, STORAGE, AND RETENTION OF DATA

After the collecting sensor data, the users can review them to verify that they do not contain sensitive information (e.g., faces in pictures or sensible locations) and judge of the appropriateness of the released information. If privacy-sensitive or inappropriate items are identified, the users can discard and delete them before the data are being reported to the application server [AAB<sup>+</sup>07]. Alternatively, the application can automatically discard the buffered sensor data unless the users review them and indicate their willingness to share the same [SBE<sup>+</sup>08].

Most of the applications discussed earlier rely on a centralized system storage managed by the application itself. The pool of sensor data is easily accessible for processing, which is advantageous for the application. However, this solution reduces the users' control over their data. Once they have uploaded their data, the users must trust the central entity not to disclose them to unauthorized third parties. Even if applications may authorize the users to delete the uploaded data or adopt retention policies favorable for the users (e.g., deleting the location information every six months by default [MRS<sup>+</sup>09]), the users do not typically receive any confirmation that data have been definitely removed from the server.

To address this issue of access control, a short-term solution may be to locally store the sensitive data on the mobile phones to prevent third parties from accessing and potentially misusing them. However, the mobile phones often suffer from resource and energy constraints, which effectively limit storage and processing capability. Moreover, this storage modality may not be appropriate to community-oriented applications, where global processing may be necessary to highlight interesting features of the data.

A solution for the secure data storage are so called Personal Data Vaults (PDVs) [MHM<sup>+</sup>10], which uncouple the acquisition of sensor data and their secure storage. PDVs are individually controlled secure data repositories, which may only be fully accessed by their owner. The owner may however choose to share information based on its time or location annotations, or provide post-processed data to external services. *SensorSafe* [CCCS11, CCS12] is another

framework that helps users in managing their data in a privacy-aware fashion. Instead of providing a unique data store accessible only by its owner as in [MHM<sup>+</sup>10], SensorSafe allows users to access multiple remote data stores by leveraging a broker. Besides, it offers similar services as the PDVs in terms of sharing. For example, users can define context-aware and behavioral-based sharing rules. Based on these rules, SensorSafe applies the corresponding obfuscation mechanisms on the stored data before their release. It further detects and notifies the users in case of rule conflicts.

Personal virtual machines also represent an alternative to local storage, designated as *virtual individual servers* [CCL<sup>+</sup>09], where the users can upload their raw data. The data are only uploaded once and may then be released to all applications the users are involved in. Different applications can be authorized to access different sets of data according to their demands. Consequently, the users maintain the control over their data and can dynamically determine potential recipients of selected sets of data. In comparison with a centralized scheme, this approach may generate additional management overhead for the users, but they retain ownership of their data and can directly control their privacy.

### 3.6 ACCESS CONTROL AND AUDIT

Depending on the application scenario, the sensing results may not only be of interest for the users, but also different stakeholders, e.g., researchers, medical staff, friends, family members, city councils, or larger public. However, the users may not be willing to share their data with all types of people within the stakeholder group and with the same granularity. Users can define the intended audience who are authorized to access their data from the user interface of many applications. They can define groups [GKN<sup>+</sup>07, GLC<sup>+</sup>08, Shio9], select persons individually [RPH<sup>+</sup>07, MLF<sup>+</sup>08, GLC<sup>+</sup>08, Shio9] or authorize everyone [GLC<sup>+</sup>08]. The users can refine their selection by specifying the nature of the data they share and may additionally define particular subsets of accessible data [RPH<sup>+</sup>07, MLF<sup>+</sup>08, Shio9]. Furthermore, they can define precise conditions (e.g., time, location, and data type) under which the data are made accessible [SBE<sup>+</sup>08]. Similarly to the selection of the sensing modalities presented in Section 3.1, these actions support the expression of the concept of appropriateness and its personalization by the users.

In order to highlight the privacy implications of sharing their data, graphic tools including maps, charts, or pictures are used to visualize the data being released and increase the users' awareness [SBE<sup>+</sup>08]. After data have been published, the users can also monitor access to the data by consulting application log files. These logs record the nature of the accessed data, the frequency of these accesses, and the identity of the people accessing them [SBE<sup>+</sup>08, Shio9, MHM<sup>+</sup>10]. Based on the results of these audits, the users control the distribution of their information and can judge of their appropriateness. If needed, they may update their access control policies to restrict or enlarge the authorization conditions in order to match their privacy preferences.

### 3.7 PRIVACY-AWARE DATA QUERY

End users may be interested in accessing data collected by other users of the participatory sensing applications. To this end, they can send a data query to the application server that contains, e.g., a location of interest. In this case, malicious application administrators may infer the location of the end users based on (1) the location from which the query is sent, and (2) the location included in the query. In the former case, the same mechanisms as applied in the reporting of sensor readings to the application server can be utilized to protect the location privacy of the end users (see Section 3.3). Despite the utilization of such a mechanisms, the location privacy of the end users may still be endangered by the location included in the query. Indeed, users likely request information about the region, in which they are currently located. Additionally, if queries can be dynamically forwarded to users located in the region of interest, the location privacy of such users need also to be protected.



In order to address these threats, three main solutions have been specially proposed for participatory sensing applications. In [DCS11], the privacy of the queries is ensured by cryptographic primitives, specially using identity-based encryption. In a first step, clients report encrypted sensor readings to the application server. Another client then sends a query to the application server. The query is also encrypted in order to protect the confidentiality of its content. At the application server, this query is blindly matched with previously received encrypted reports. As a result, the application administrators have neither access to the content of the reports nor the queries. In case the query matched existing reports, the corresponding reports are transmitted to the requester who is the only one able to decrypt them and access the original sensor readings.

Instead of relying on cryptographic solutions, the following approaches aim at limiting the location information provided to the application server while still supporting the answers to potential queries. In [BFV12], the authors show that the transmission of the  $k$  last positions visited by the users to the application server is sufficient to provide a satisfying coverage of a region of interest in order to respond to the corresponding queries. Another approach proposed in [KS11a, KS11b] is based on the dynamic and decentralized distribution of the regions of interest between users. The clients first share the locations of interest among them by computing their Voronoi cell without the intervention of any central entity. Next, they define a cloaked area containing their location and at least  $k - 1$  other users in order to protect their privacy. They further cloak the smallest radius of their Voronoi cell by replacing it by the maximum (smallest) radius among the  $k - 1$  users. This prevents malicious application administrators from identifying users based on this radius. The clients report both cloaked location and radius to the application server that finally assigns the locations of interests to each user accordingly.

### 3.8 SUMMARY

In this chapter, we have surveyed the current state-of-the-art in privacy protection mechanisms specially tailored to participatory sensing applications. Our survey has first shown that users are not involved in all privacy-preserving mechanisms. As summarized in Figure 3, users can mainly indicate their preferences in terms of sensing and released location granularity. They can review the collected data, manage their storage, and define the related access control before their release to potential end users. With the exception of these approaches, users are not always involved in privacy decisions for mechanisms protecting their privacy while interacting with the participatory system. As a result, most of the proposed mechanisms are controlled and pre-configured by the applications and not by the users.

Secondly, most proposed schemes rely on a centralized entity, either the application server or a trusted party. Hence, users must trust this entity to (1) protect their privacy according to their preferences and (2) not to disclose their data to unauthorized parties. Indeed, users loose the control over their data once they either upload the sensor readings to the application server or transmit their location information to a trusted party. While users can configure settings for data release, they have no guarantees that the application server will adhere to them.

---

There are no problems, only solutions.

J. Lennon

---

Based on the survey of existing countermeasures to privacy threats presented in Chapter 3, we have identified two main shortcomings in Section 3.8, namely the prevalence of both application-controlled and centralized privacy-preserving schemes. In this section, we discuss both identified shortcomings and specially highlight the arising research questions that are addressed throughout this thesis.

#### 4.1 APPLICATION-CONTROLLED TO USER-CONTROLLED PRIVACY-PRESERVING SCHEMES

As a first issue, our survey of existing privacy-preserving schemes has shown that numerous schemes are fully configured by the participatory sensing applications and users are hence not involved in privacy decisions at all. Thus, users are unable to (1) express their privacy preferences and (2) exert any control over how their data are released to and accessed by applications. Privacy research has shown that the notion and perception of privacy are strongly individual [Wes67] and furthermore based on the user's context [FHHK<sup>+</sup>11] and cultural background [LS93, GHW<sup>+</sup>11]. The combination of these factors thus has a strong impact on the individual perception of data sensitivity, and we believe that no universally valid configuration exists for large and heterogeneous user bases.

By selecting general settings valid for all users, application administrators express and impose their own vision of privacy on the users. If this vision does not match the privacy conception of the users, users may (1) opt out if they judge the provided protection to be insufficient, or (2) provide less information to the application than if they would have set the preferences themselves (if their conception of privacy is less restrictive than the one applied).

Participatory sensing applications depend on user-provided data. The users' reluctance to contribute would diminish the impact and relevance of sensing campaigns as well as the benefits to the users. In order to foster the contribution of users, we therefore believe that privacy-preserving mechanisms should cover more than one conception of privacy. In particular, users should be able to configure these mechanisms according to their own conception.

By not involving users in privacy decisions, existing mechanisms not only prevent customization of privacy settings, but also prevent users from exerting control over their data. In the few studied solutions that permit individual privacy settings, users can mainly control the type of sensor readings provided to the application, the degree of granularity, the spatiotemporal context, and the data recipients (cf. Chapter 3). In most cases, users have no control over the mechanisms applied on their data by the application server or trusted parties; these mechanisms remain hidden from them. Control and transparency are, however, pivotal components in the acceptance of privacy-enhancing technologies by potential users [CsS<sup>+</sup>05, ACC<sup>+</sup>05, HFHPBo7, GHW<sup>+</sup>11]. In orthogonal domains to participatory sensing, experiments have further demonstrated that providing control to users over their data and privacy protection increase their trust in the system [FHP04, PFHD<sup>+</sup>05, GHW<sup>+</sup>11]. As a result, we believe that increasing user control on applied privacy-preserving mechanisms in participatory sensing would similarly increase the trust of the users in the system and thus, foster their contribution to the application. As a result of this discussion, we identify the following open issues in current participatory sensing systems:

- ▷ How can users of participatory sensing applications control their privacy?

- ▷ Which parameters can be controlled by users?
- ▷ What is the impact of this control on the function of the application and its parameters, such as overhead and latency?

Investigating these questions paves the way to the inclusion of users in the loop. Beyond empowering users to exert control over their privacy, such investigation is important to determine the consequences of this control on the application and its applicability in real-world participatory sensing deployments. The contributions of this thesis, as detailed in Chapters 5 to 7, answer the identified research questions by proposing and evaluating novel user-controlled privacy-preserving schemes that can be tailored to the users' privacy preferences.

#### 4.2 CENTRALIZED TO DECENTRALIZED PRIVACY-PRESERVING SOLUTIONS

Secondly, as highlighted in our preceding survey (cf. Section 3.8), most of existing solutions are organized around a central entity, either the application server or a trusted party, that ensures the privacy protection of the users. In this scenario, all sensor readings collected by the users are first transmitted to the central entity that applies privacy-preserving schemes in order to remove sensitive information from the uploaded data. As a result, application and/or trusted party administrators have direct access to the reported sensor readings. They are also responsible for the selection of the applied privacy-preserving scheme(s) and ensure the distribution of the processed sensor readings to potential end users. As users lose the control over their data once they have been uploaded, they must trust the central entity to protect their privacy according to their preferences and not to disclose their data to unauthorized parties. In most solutions, the administrators of these central entities are assumed to be honest and respect as well as protect the privacy of the users. However, users have no guarantees about the honesty of the administrators, which may be interested in their data for various purposes, such as simple curiosity or willingness to commercially exploit the contributed data. Depending on their interest(s), administrators can violate the privacy of the users to different degrees and finally break the trust of the users. Additionally, such centralized architectures present a single point of failure, making them vulnerable to malfunctions and malicious external attacks that can reveal or damage the entire data storage. Their vulnerability has been demonstrated multiple times in recent years through the leakage of user names and passwords stored in central repositories, such as in Sony's PlayStation Network [AS11] and Yahoo Voices [Gro12].

Based on this discussion, we identify the following open issues:

- ▷ How can the dependency of the users on trusted entities be reduced?
- ▷ How to support the functions of participatory sensing applications, such as reputation systems, while reducing the dependency to trusted entities?
- ▷ Which privacy processing functions can be shifted to the users' mobile phones? Which one(s) cannot?

The investigation of these research questions is important to empower users with additional control over their data, while simultaneously taking into account the existing underlying architecture of participatory sensing applications. By answering these questions and adopting a threat model, in which application administrators and trusted parties can show malicious behavior, we believe that the resulting solutions are better adapted to the conditions of real-world participatory sensing deployments and offer a better privacy protection to the users of such applications.

#### 4.3 SUMMARY

In this chapter, we have defined our problem statement based on our literature review conducted in Chapter 3. Our problem statement follows two primary research directions.



Firstly, we aim at providing solutions in which users are in control of their privacy protection and can adapt the underlying mechanisms to their individual privacy preferences. Secondly, our objective is to increase user control by moving from existing centralized solutions towards decentralized architectures in order to reduce the dependency of users on application and trusted party administrators. As a result, the contributions of this thesis are in line with both identified research directions and answer the questions raised in this chapter.



## PATH JUMBLING: A PRIVACY-PRESERVING COLLABORATIVE SCHEME FOR PATH HIDING

---

Alone we can do so little;  
together we can do so much.

---

H. Keller

We have shown that most existing schemes to protect privacy in participatory sensing rely on either the application or a third party. For example, spatial cloaking requires users to report their exact locations to a third party that generates the cloaked values [HKH09]. Users therefore need to entrust the protection of their privacy to these entities. Besides, we have shown that users are not involved in most privacy decisions taken in these schemes, despite the individual and personal nature of privacy. In order to address both of these issues, we propose the concept of path jumbling [CGR<sup>+</sup>11], in which users mutually preserve their privacy in a decentralized fashion and can parametrize the proposed scheme according to their individual preferences. In our scheme, users physically exchange collected sensor readings with other users at opportunistic encounters. As the sensor readings are annotated with spatiotemporal information, the exchanges aim at breaking the association between the users' identities and the locations visited during the collection of the sensor readings and thus, preserve the location privacy of the users.

We first define our system and adversary models in Section 5.1, before detailing our concept in Section 5.2. In the description of our concept, we specially analyze the design space, define design criteria, and present our design decisions. Next, we evaluate our design decisions using real-world GPS traces of human mobility. Our evaluation is two-fold. In Section 5.3, we conduct a quantitative analysis of our design decisions in comparison to our design criteria, while we measure the resilience of our concept against potential adversaries in Section 5.4. Finally, we survey direct related work in Section 5.5, before summarizing our results in Section 5.6.

### 5.1 SYSTEM AND THREAT MODELS

#### 5.1.1 System Model

For our system, we assume participatory sensing applications without real-time constraints for data delivery. Examples include monitoring noise pollution [RCK<sup>+</sup>10] and road conditions [MPRo8] as well as documenting personal diets [RPH<sup>+</sup>07]. In these applications, the users carry mobile phones equipped with embedded sensors, wireless interfaces, and positioning systems (e.g., GPS, Wi-Fi, or cellular network based triangulation [LCC<sup>+</sup>05]). The mobile phones autonomously collect sensor readings (e.g., sound samples, pictures, and accelerometer data). However, the users can control the activation/deactivation of the sensing function and its sampling period. Each sensor reading is stamped with the collection time and location information to form the following triplet  $T = \langle t, l, s \rangle$  with  $t$ : time,  $l$ : location,  $s$ : sensor reading, which may be present in the form of vectors (like 3-axis accelerometer readings) or scalar value (such as noise level in decibels). Additional processing can be locally applied to the sensor readings to extract features and/or avoid the disclosure of sensitive information. For example, possibilities include extracting the noise level from the collected sound samples, or obfuscating recorded conversations in applications monitoring noise pollution.

The triplets are then autonomously reported to the application server. We assume that the users can configure reporting settings by, e.g., selecting the desired reporting frequency and particular reporting locations or choosing to only report when they have access to a free Wi-Fi

connection. The application server is able to establish a link between the reported triplets and the users who reported them. The establishment of this link can be based on either explicit identifiers, such as user ID and pseudonyms, or the analysis of reporting meta-data to, e.g., infer the location from the used IP addresses. The application can utilize the established links to assign reputation to the users [HKH10b], as from the application perspective the triplets are assumed to be reported by the users who collect them.

### 5.1.2 Threat Model

We assume two kinds of adversaries to our system: (1) malicious application administrators and (2) malicious users. The motivations of the adversaries include simple curiosity and willingness to harm or commercially exploit the disclosed information. Both types of adversaries are discussed in more detail in the following.

Malicious administrators are a common threat to the privacy of the users in participatory sensing applications, since they have direct access to the triplets reported by the users and stored on the application server. In absence of privacy-preserving mechanisms locally applied on the mobile phones, the triplets contain information about the locations visited by the users and disclose the paths they followed. This may provide insights about the users' whereabouts and lifestyle. For example, the sensor readings collected while users are walking from their offices to their homes might reveal their exact paths as well as their start and end locations. Since the collected sensor readings are reported and stored on a central application server, they can be exposed to the following threats to their privacy:

- ▷ Curious administrators who attempt to gain further information about the participants by analyzing their sensor readings.
- ▷ Intentional disclosure to untrusted parties and/or misuse of the sensor readings and derived information by malicious application administrators.
- ▷ Application of inefficient privacy-preserving mechanisms by the administrators, potentially leading to privacy breach when sharing the sensor readings with third parties.
- ▷ External attacks directed at the server to fraudulently access the data stored.

In the following, we assume a honest-but-curious adversary model, in which the application administrators attempt to passively breach the privacy of the participants, but runs the system normally and faithfully. This means that the application administrators focus on the data reported by the participants to the application server in order to breach their privacy. They, however, do not launch active attacks (such as collusions with malicious users) to obtain further information.

As an artifact of the collaborative nature of the path hiding mechanism, users can also become adversaries. Malicious users can launch the following attacks:

- ▷ They can attempt to breach the privacy of other users by directly accessing the information included in the triplets exchanged during encounters.
- ▷ They can stay in crowded areas to collect as many triplets as possible.
- ▷ They can either drop exchanged triplets or create false triplets to be exchanged in order to alter the results consolidated at the server side and perturb the function of the collaborative path hiding mechanism. Note that we address this particular adversary scenario in Chapter 6.

We further assume that adversaries follow the Dolev-Yao threat model [DY83], i.e., they are able to listen to all communication, fabricate, replay, and destroy messages. They are, however, not able to compromise cryptographic mechanisms.

## 5.2 THE PATH JUMBLING CONCEPT

In this section, we first provide an overview of the path jumbling concept. Next, we analyze the design space for both exchange and reporting strategies, before presenting our design decisions.

### 5.2.1 Overview

The objective of our concept is to remove the association between the spatiotemporal context (i.e., time and location) at which the sensor readings were taken and the identity of the users in order to protect their privacy. The spatiotemporal context reveals the visited locations and paths followed by the users during the sensing process, thus providing insights about the users' lifestyles and whereabouts. In our approach, users mutually preserve their privacy in a decentralized fashion. Inspired from mix networks [Cha81], users exchange previously collected sensor readings when they physically encounter each other. Before their exchange, the triplets are individually encrypted using the public key of the server to prevent other users from accessing their content.

In order to illustrate our concept, we consider the example in Figure 4. We assume that users A, B, and C follow the paths illustrated in Figure 4a while they collect triplets and meet according to the timeline represented in Figure 4b. In this example, A first meets C and exchanges his previously collected triplets with him. The selection of the triplets is determined by the exchange strategy. Here, we assume that A and C exchange all triplets they collected up to the moment of their encounter. Note that different exchange strategies can be envisaged, as discussed in Section 5.2.2. After their meeting, A and C continue to collect triplets while they are continuing along their routes. When A meets B, both exchange their triplets (including those already exchanged by A) according to the exchange strategy. After their meeting, they collect triplets until reaching their final destination where they terminate the sensing process.

We further assume in the presented example that the users configure their reporting strategy to a daily upload of triplets. Note that the users can select other reporting strategies as detailed in Section 5.2.3, which have an impact on the achievable privacy protection. By applying our mechanism, the users report triplets partially collected by themselves and partially exchanged with other users to the application server. Based on these triplets, the application assumes that user A walked from B's home to the bank. The user B traveled from the cinema to the hospital, and the user traveled C from A's office to his home, as depicted in Figure 4c.

As a result, swapping a subset of the data samples removes the association between the sensor readings and the identity of the users who collected them. Consequently, the sensor readings do not reveal the actual paths followed by each user, but instead jumbled paths, which cannot be easily reconstructed by a server as shown in Section 5.4.

### 5.2.2 Exchange Strategies

As mentioned in Section 5.2, different exchange strategies can be applied to jumble triplets between users. In this section, we analyze the design space and select different strategies to investigate their impact on the performance of our concept in Sections 5.3 and 5.4.

#### *Design Space Analysis*

Our analysis primarily concentrates on the selection modality of triplets to exchange, both locally collected and received from other users. This includes a discussion about the number and the selection of triplets that will be exchanged. We consider the following design alternatives:

- ▷ **PARTIAL VS. COMPLETE EXCHANGES** The triplets can be exchanged either partially or completely upon physical encounters at participants. A partial exchange is defined

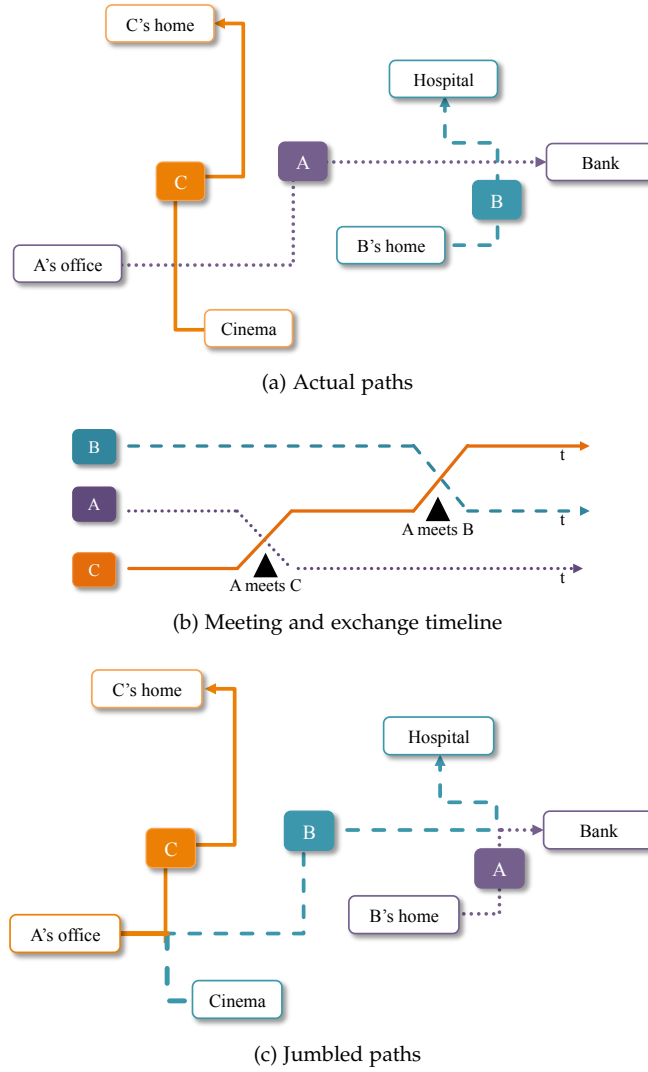


Figure 4: Meetings and exchanges of the users A, B, and C

as the transmission of a random subset of all collected and already exchanged triplets. Opposed to the traces resulting from the partial exchange of triplets, paths generated by the latter alternative are realistic (see example in Section 5.2), as if captured by a single real person. When such paths are reported to the application after the jumbling process, the users' actual traces are obfuscated by realistic and coherent substitutes. The application server can however identify potential exchanges between users based on the spatiotemporal distribution of the triplets, as the users' paths overlap at encounters. Since users exchange all triplets at each encounter, the application server would be able to reconstruct all jumbled paths by simply switching the subpaths. Instead of exchanging all triplets at each encounter, we propose that users exchange all of them with a certain probability in order to prevent the application from reconstructing the jumbled paths straightforwardly. By choosing a probability of 0.5, the application server has therefore a  $0.5^n$  chance of correctly identifying the original paths of the users after  $n$  encounters.

In comparison, partial exchanges may lead to incoherent traces, since the exchanged triplets may not necessarily have been collected in the same area. Malicious application administrators may hence be able to recognize the exchanged triplets because of their outlying location. Besides, users will often report a higher percentage of own triplets to the server, as they only partially exchange them. As a result, users may reveal

more information to the server about themselves than when complete exchanges are performed.

- ▷ **INDIVIDUAL VS. CONSECUTIVE TRIPLETS** If the aforementioned partial strategy is applied, the triplets to exchange can be chosen either randomly, such that they represent a collection of sparse locations, or by selecting coherent path segments based on consecutive triplets. The exchange of partial triplets may, however, form unrealistic paths as the collected and exchanged triplets may have been collected in unconnected areas and at different times of the day. This can result in improbable covered distances between triplets assumed to be consecutive, which can be detected by the application server.
- ▷ **SYMMETRIC VS. ASYMMETRIC EXCHANGES** Another design dimension to explore is the reciprocity in the amount of triplets exchanged during an encounter. A symmetric exchange supports the collaborative nature of our concept, as the users benefit from similar exchange conditions and potential reporting overhead can be distributed between them in a fair manner. Instead of choosing a predetermined value of triplets to be users by all users, a negotiation phase is used before the actual data transmission to agree on the amount of exchanged triplets. However, this negotiation must take into consideration that malicious users may be willing to exchange a large amount of data with other users to gather as much information as possible. In comparison, an asymmetric exchange allows each user to individually determine how many triplets she exchanges, implying that users can receive more triplets than they exchange. These users will thus report more triplets to the server introducing additional overhead for them.

The above discussions highlight that each alternative presents advantages as well as drawbacks with regard to potential information disclosed to the server and introduced overhead. Tradeoffs between advantages and drawbacks must thus be found.

### *Design Decisions*

Based on the results of the design space analysis, we select the following exchange strategies for our further analysis, which cover different combinations of the above design alternatives:

- ▷ **REALISTIC EXCHANGE STRATEGY** The users exchange their entire set of collected/exchanged triplets at each meeting with a certain probability. Let us go back to our example shown in Figure 4 and assume that users A and C collected six and three triplets before their meeting, respectively. A hence receives the three triplets from C, and C receives the six triplets from A, as illustrated in Figure 5a. This strategy exchanges consecutive triplets, which form realistic path segments (see Section 5.2). The exchange can be asymmetric, if the amounts of collected/exchanged triplets differ between both users.
- ▷ **RANDOM-UNFAIR EXCHANGE STRATEGY** Each user independently and randomly determines the amount of triplets she wants to exchange. In Figure 5b, A exchanges five of six triplets, while C exchanges two of three triplets. Moreover, each triplet to exchange is selected randomly. In our example, A selects the triplets  $A_1$ ,  $A_2$ ,  $A_3$ ,  $A_4$ , and  $A_6$ , whereas C selects the triplets  $C_1$  and  $C_3$ . In comparison to the realistic strategy, the exchanges are mostly partial and involve triplets selected individually. The exchanges can be asymmetric, depending on the amount of exchanged triplets.
- ▷ **RANDOM-FAIR EXCHANGE STRATEGY** The users agree on a common amount of  $n$  triplets to exchange at each meeting. Each user advertises the amount of triplets available for exchange. In Figure 5c, A and C advertise six triplets and three triplets, respectively. The amount of exchanged triplets is determined randomly between one and the minimum of both advertised values. In our example, a value of one triplet is chosen. Then, A and C randomly select the triplet to exchange. In comparison with the random-unfair strategy, the fair variant ensures the symmetry of each exchange that fairly jumbles the triplets and equally distributes the reporting overhead between the users.

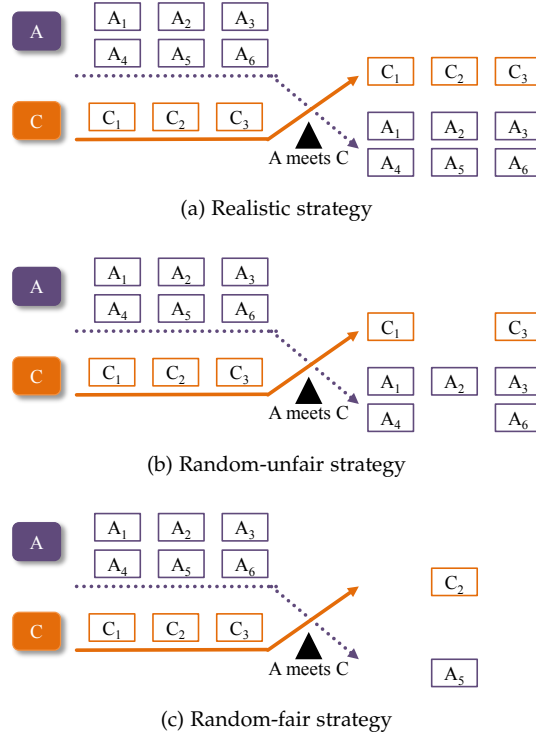


Figure 5: Comparison of the exchange strategies

Furthermore, we complete these strategies by introducing additional features to prevent consecutive exchanges with the same users and unidirectional exchanges of triplets. The first feature only allows users to exchange triplets with the same user when they have exchanged data with at least  $x$  other users between two encounters. A malicious user can thus not easily recover a full collection of triplets by simply following targeted users and constantly exchanging triplets with them. Without this feature, already exchanged triplets may be exchanged again and return to the users who collected them. This would lower the jumbling degree of the triplets, and hence the privacy benefits of our approach. Moreover, we introduce a tit-for-tat mechanism where triplets are exchanged alternatively to ensure that malicious users cannot receive an entire set of triplets without providing any triplet in exchange. Additionally, both users have an equivalent opportunity to exchange triplets even in case of early abortion of the exchange due to technical reasons or divergent user mobility.

### 5.2.3 Reporting Strategies

In addition to the exchange strategy, different variants can be envisaged to report the triplets to the application server. In this section, we analyze and discuss design alternatives before presenting our design decisions.

#### Design Space Analysis

The privacy protection provided by our approach depends on the meeting pattern of the users. In extreme cases, users may not be able to exchange triplets with other users during long time periods. The non-jumbled triplets can be either reported to the application server or stored until the next meeting. Reporting these triplets to the server would reveal the original paths followed by the users, and hence breach their privacy. In contrast, waiting until the next meeting to report the triplets would introduce additional delays for the application. Within the scope of this analysis, we thus investigate the tradeoff between privacy and latency by considering the following reporting strategies:



- ▷ **TIME-BASED STRATEGY** The triplets are periodically reported to the server. The application is thus ensured to timely receive triplets. However, the triplets can be reported without having been jumbled in absence of encounters during the considered period.
- ▷ **EXCHANGE-BASED STRATEGY** The triplets are reported to the server after a certain number of meetings. The reporting latency is thus determined by the frequency of the meetings. Although this strategy ensures that the triplets have been jumbled before their report, no guarantee is provided on the achieved degree of jumbling. For example, only one of the reported triplets may have been jumbled with another user.
- ▷ **METRIC-BASED STRATEGY** The triplets are only reported to the server after reaching metric-based thresholds, such as a given number of triplet exchanges. This strategy thus guarantees the users that their privacy is respected to a degree defined by the threshold. However, these thresholds may increase the latency between two reports, as multiple meetings may be necessary to reach them.

The discussed reporting alternatives highlight that high privacy protection and low latency for the application are difficult to combine. A tradeoff must thus be found between both parameters, especially as users may refuse to contribute to the application if their privacy is not protected. To the interests of the application, these contributions must be encouraged, meaning that strategies with guaranteed privacy should be preferred, despite the introduction of additional latency for the application.

#### *Design Decisions*

Based on the previous discussion, we select different reporting alternatives in order to investigate their actual impact on the performance of our concept. Note that the parameterization of the strategies has been influenced by the real-world dataset used for the evaluation (see Section 5.3).

- ▷ **HOURLY AND DAILY STRATEGIES** The triplets are reported hourly and daily, respectively. Both time-based strategies may report non-jumbled triplets in absence of meetings. In comparison with the hourly strategy, the daily strategy offers a longer period during which meetings may occur.
- ▷ **1-EXCHANGE STRATEGY** The triplets are reported after each exchange. We introduce a random waiting period before the upload to prevent the server from identifying the users who exchanged their triplets by analyzing simultaneous reports.
- ▷ **JUMBLING-BASED STRATEGY** The triplets are reported if the percentage of jumbled triplets (i.e., collected by others) reaches a given threshold. We choose the following thresholds: 25%, 50%, and 75%. This metric-based strategy allows controlling that the triplets have been sufficiently jumbled. A high percentage indicates that only few triplets were collected by the users themselves and their reporting discloses thus little information about the paths actually followed by them.
- ▷ **DISTANCE-BASED STRATEGY** The triplets are reported if the average distance between each location of the actual and jumbled paths is above a given threshold. We select three thresholds: 1 km, 2 km, and 5 km. Similarly to the former strategy, the distance-based strategy provides an estimation of the path. If the distance is small, the jumbled path remains in proximity of the actual path and may still contain sensitive locations. If the distance is large, fewer insights about the users can be inferred.

In summary, we have chosen different exchange and reporting strategies that can be selected based on users' preferences in terms of privacy protection. Both the performance of the chosen strategies and the resulting degree of privacy protection are examined in the next sections.

### 5.3 PERFORMANCE IN EXCHANGING AND REPORTING TRIPLETS

In this section, we describe the performance evaluation of the selected exchange and reporting strategies detailed in Section 5.2.2 and Section 5.2.3, respectively. We first discuss the characteristic properties of the utilized dataset and detail our simulation settings. Subsequently, we present the results of our evaluation and highlight particular findings.

#### 5.3.1 *Simulation Setup and Method*

Our evaluation is based on the GPS traces from the GeoLife project ([ZLC<sup>+</sup>08], [Mic13]). In this real-world deployment, the users carried GPS-enabled devices to monitor their location. We extend the initial scenario to a participatory sensing application by assuming that a triplet was collected at each monitored location. In total, 178 users contributed to this project between April 2007 and October 2011. This resulted in the collection of more than 17,500 trajectories mainly located in Beijing, China with a sampling frequency between one and ten seconds depending on the user. From this dataset, we have selected 97 users, which had at least met one other user, i.e., having been in a range of 20 m for at least 8 s, and we observed their mobility and meeting pattern during a period of 24 hours. The meeting distribution of the selected users is depicted in Figure 6a, while their respective number of collected triplets is presented in Figure 6b. Note that the difference between the numbers of collected triplets is due to the possibility for the users to select the collection frequency as well as activate/deactivate the sensing function. Among the 97 users, we selected the following users who represent the extreme and average cases in terms of meetings: User 19 (one meeting), user 55 (three meetings), and user 97 (17 meetings).

We simulated the exchange and reporting of triplets based on the GPS traces in order to evaluate our approach. When the users were in physical proximity, they exchanged their triplets according to one of the exchange strategies (see Section 5.2.2). For the realistic strategy, we choose a probability of exchange equal to one in order to compare all exchange strategies on the same basis, i.e., the same number of exchanges. The following results show therefore the upper bound for the realistic strategy. Each user was only able to exchange with the same user again, if she met three other users in between in order to prevent constant exchanges between the same users as discussed in Section 5.2.2. Although this value has been selected based on the characteristics of the dataset, it could also be easily determined dynamically in a real-world deployment. Besides, the users reported the triplets to the application server with one of the selected reporting strategies (see Section 5.2.3). We therefore conducted a cross-evaluation of both exchange and reporting strategies, where each possible combination of strategies is evaluated using the following metrics:

- ▷ **JUMBLING DEGREE** It measures the average percentage of reported triplets having been jumbled with other users. A high percentage thus indicates that only few triplets collected by the users themselves are reported, meaning that little information about the users' paths is disclosed. This metric hence provides insights about the level of obfuscation achieved at the time of the reporting to the server.
- ▷ **DISTANCE** It estimates the average distance between the actual path followed by the users and the path resulting from the exchange. A small distance indicates that the reported path remains in proximity of the actual path. Even when the distance is large, the reported path may however still reveal sensitive locations visited by the users.
- ▷ **OVERHEAD** It compares the average number of triplets having been reported after jumbling with the number of triplets having been collected. It thus measures the reporting overhead caused by our approach. A positive factor/percentage means that the users need to report more triplets to the server than they collected themselves after applying our mechanism.

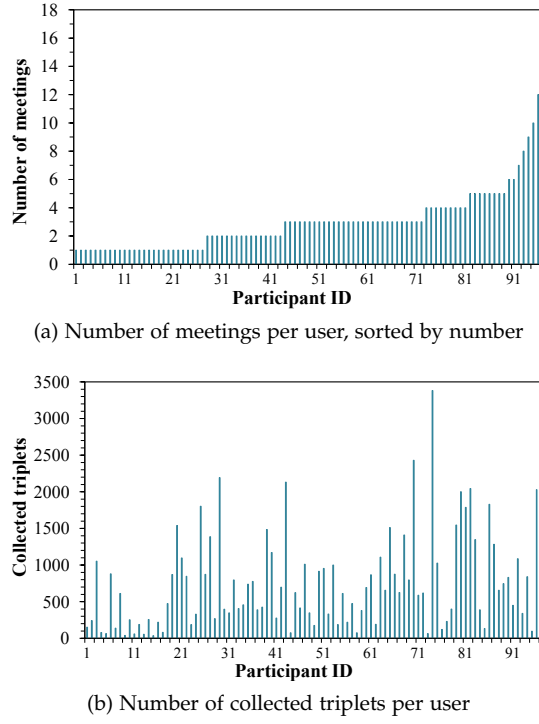


Figure 6: Distribution of meetings and triplets for the 97 users over 24 hours

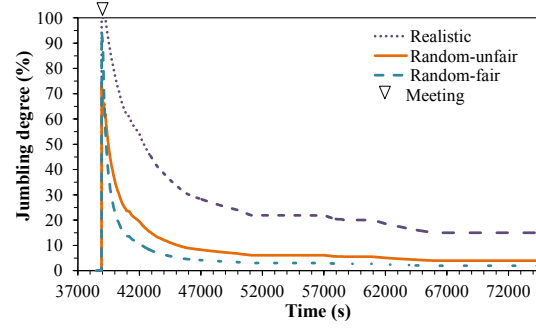
### 5.3.2 Results

In this section, we compare and evaluate the impact of the selected exchange and reporting strategies on the aforementioned metrics. Although the results of our evaluation are only representative for the given dataset, our evaluation shows the feasibility of our approach under realistic conditions.

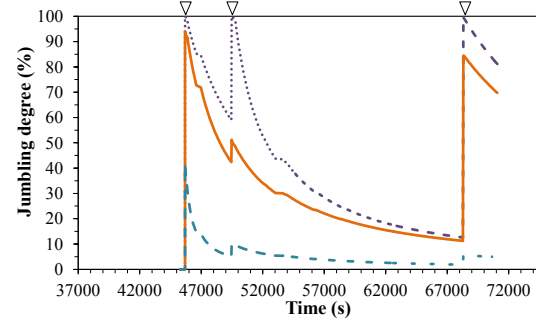
#### *Jumbling Degree*

We consider the realistic, random-fair, and random-unfair exchange strategies and assume first that the triplets are reported at the end of the sensing process. In particular, we examine the results for three representative users 19 (one meeting), 55 (three meetings), and 97 (17 meetings) introduced in Section 5.3.1 in order to investigate the impact of the number of meetings on the performance of our approach. Figure 7 illustrates the temporal evolution of the jumbling degree for each user. Note that the users have different sensing periods, as they were able to activate/deactivate the sensing function individually. The figure highlights that the realistic strategy ensures the highest jumbling degree, as all triplets are exchanged at each meeting. For the realistic strategy, the jumbling degree hence reaches 100% at the time of each meeting and then progressively decreases when users collect triplets until the next meeting. In comparison, both random-unfair and random-fair strategies reach lower jumbling degrees due to partial exchanges of randomly selected triplets. For example, the jumbling degree of user 55 is equal to 94% for the random-unfair strategy and 42% for the random-fair strategy at  $t=45,678$  s (time of his/her first meeting), respectively. Moreover, the random-unfair strategy globally shows a higher jumbling degree than the random-fair strategy since more triplets are exchanged at each meeting with the random-unfair strategy.

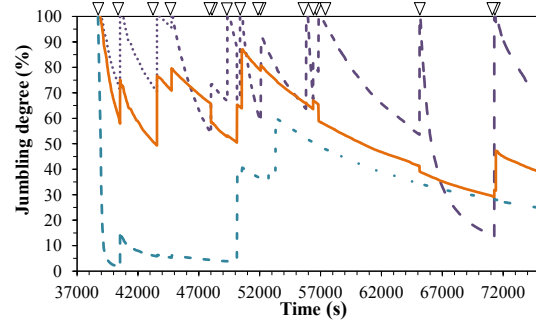
We next analyze the combination of exchange and reporting strategies for the entire dataset, i.e., all 97 users. For each combination, we calculate the minimal, mean, and maximal jumbling degree of the triplets reported to the application server. Table 3 shows the obtained results, which can be summarized as follows. The realistic strategy combined with all reporting strategies shows a higher jumbling degree than the random-unfair and random-fair strategies.



(a) User 19 (one meeting)



(b) User 55 (three meetings)



(c) User 97 (17 meetings)

Figure 7: Jumbling degree over time for selected users with different numbers of meetings

Except for the time-based reporting strategies, the jumbling degree reaches 100%, meaning that all triplets were jumbled before their reporting to the application server, even for users with a single meeting only. This implies that the paths are protected independently of the selected reporting strategy and even if the users meet once only. In addition to the high jumbling degree, this exchange strategy allows the application to timely receive the triplets, as the thresholds triggering the report can already be reached after a unique meeting, while multiple meetings may be needed in the case of the random-unfair and random-fair strategies.

In general, the random-fair strategy obtains lower jumbling degrees than the random-unfair strategy due to its fairness constraint, as illustrated in Figure 8. For both random-unfair and random-fair strategies, the jumbling-based reporting strategies show higher jumbling degrees than with other reporting strategies. Note that the minimum values of the jumbling degree are at least equal to the threshold of the applied jumbling-based strategy. For example, the minimal jumbling degree obtained by applying the random-unfair exchange strategy and the 25%-variant jumbling-based reporting strategy is equal to at least 25%. The jumbling-based reporting strategies therefore guarantee minimal jumbling degrees, which is not the case for the other reporting strategies. This is also clearly seen, as the time-based reporting strategies

Table 3: Strategy comparison: Jumbling degree, distance, and overhead

			METRICS				
			JUMBLING DEGREE (%)			DISTANCE (km)	OVERHEAD (%)
Exchange strategy	Reporting strategy	Threshold	Mini- mum	Mean	Maxi- mum	Median	Median
Realistic	Time-based	hourly	0	17	49	2	8
		daily		59	100	5	7
	Exchange-based	1	100	100		4	-8
	Jumbling-based	25%					
		50%					
		75%					
	Distance-based	1 km				5	0
		2 km					-6
		5 km					8
Random-unfair	Time-based	hourly	0	8	32	2	5
		daily		28	88	5	7
	Exchange-based	1	36	97	4	-4	
	Jumbling-based	25%	26	61	96	5	-26
		50%	53	76	97		-46
		75%	75	89	99		-78
	Distance-based	1 km	0	36	97		-3
		2 km		30	96		-9
		5 km	1	49	98	9	-27
Random-fair	Time-based	hourly	0	2	11	2	0
		daily		10	39	5	
	Exchange-based	1	22	65	4		
	Jumbling-based	25%	26	42	76	6	
		50%	51	62	79	5	
		75%	76	79	80	9	
	Distance-based	1 km	0	21	65	7	
		2 km		19	63	4	
		5 km	1	21	80	7	

show particularly low jumbling degree and do not provide any guarantee, as the triplets are reported even in absence of meetings.

As compared to the jumbling-based reporting strategies that guarantee a predetermined maximal jumbling degree, the remaining reporting strategies achieve variable maximal jumbling degrees, which range from 65% to 100%.

#### Distance

Similarly to the evaluation of the jumbling degree, we consider each combination of exchange and reporting strategies and calculate the distance between the collected and the reported triplets for all users. The second column to the right of Table 3 presents the obtained median distances. Note that the obtained values depend on the mobility pattern of the users. In isolation, these values only provide limited insights, while their comparison allows to evaluate the performance of the combined exchange and reporting strategies for preserving the path privacy. In average over all reporting strategies, the median distance between positions on the real and jumbled path obtained by the realistic strategy is 4 km, while it is 5 km for both

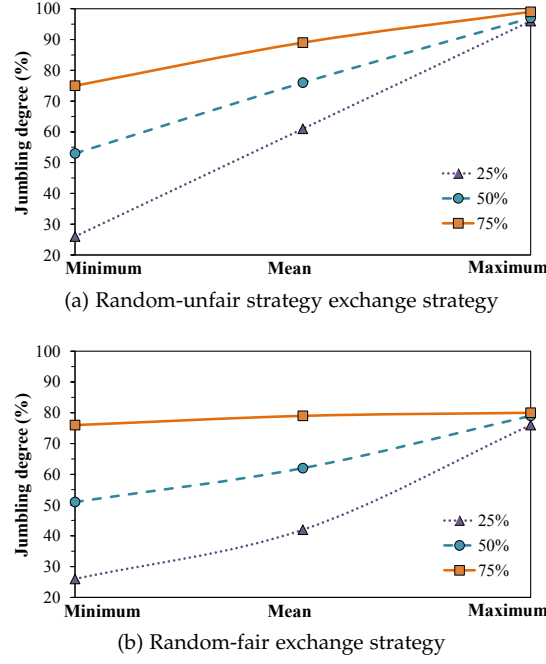


Figure 8: Jumbling degree of the jumbling-based reporting strategies

random-unfair and random-fair strategies. The impact of the different exchange strategies is thus only slightly distinguishable. The analysis of the different reporting strategies shows that there is no major difference in terms of distance between them. However, the distance-based reporting strategies ensure that the reported triplets at least reach their respective distance threshold. The users are therefore guaranteed that the reported triplets present a distance to the actual path greater than the predetermined threshold and that their privacy is respected to a greater extent. The hourly strategy performs poorly, as the triplets are often reported even in absence of meetings.

#### Overhead

Except for the random-fair strategy, the exchange strategies can influence the amount of triplets to report, as the users can receive a lower, equal, or higher amount of triplets than they provide. The difference can thus increase, leave unchanged, or reduce the reporting overhead.

In a first step, we evaluate the potential reporting overhead introduced by the exchange strategies combined with the daily reporting strategy. Figure 9 illustrates the distribution of the reporting overhead factor. The realistic strategy introduces a larger overhead factor than the random-unfair strategy. The overhead remains negative for 44 users with realistic strategy and 40 users with the random-unfair strategy. This implies that these users report fewer triplets than collected by themselves with our approach. On the other hand, the remaining users report more triplets than they collected with a factor reaching up to 55 for the realistic strategy and 40 for random-fair strategy. On a network scale, the number of reported triplets however remains identical.

Secondly, we examine the median reporting overhead introduced by the exchange strategies in combination with different reporting strategies. The results are summarized in the third column to the right of Table 3. By design, the random-fair exchange strategy does not introduce/shift any overhead. The comparison of both realistic and random-unfair strategies shows that the overhead median is -4% on average for the realistic strategy and -20% on average for the random-unfair strategies. This result confirms and extends the prior finding: The overhead introduced by the realistic strategy is generally higher than with the random-unfair strategy. The overhead difference, however, depends on the random value that determines the number of triplets to exchange in the random-fair strategy. Moreover, the reporting

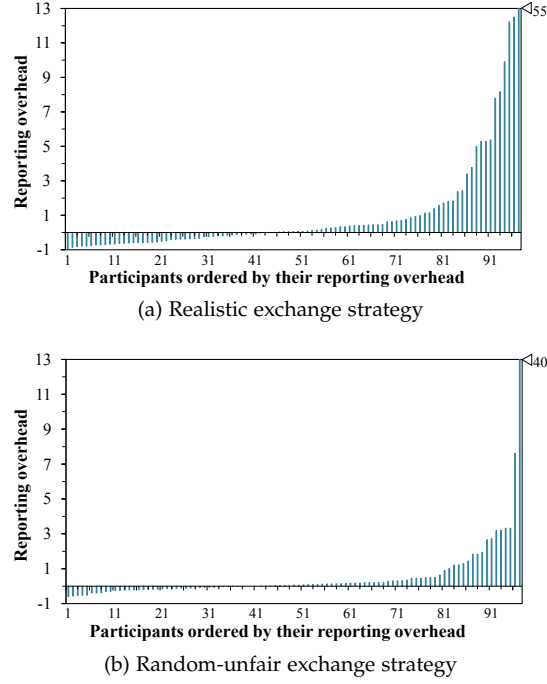


Figure 9: Distribution of the additionally introduced reporting overhead factor for the daily reporting strategy

performance is also influenced by the random selection of the triplets, as the thresholds can be either exceeded or not, depending on the selected triplets. Note that the median reporting overhead is negative in most cases. Intuitively, one would expect its value to globally be zero, as the triplets given by one user are received by another. However, one needs to remember that we only consider the triplets that are actually reported (and not only exchanged) in Table 3. If the reporting conditions are not fulfilled, the triplets are not reported and not taken into account in the calculation of the reporting overhead.

In summary, the above evaluation shows that the realistic exchange strategy provides the best results in terms of their jumbling degree. Except for the time-based reporting strategy, the jumbling degree reaches 100% even after a single meeting, while additional meetings increase the mixing of the triplets among the users even further. As a single meeting is sufficient to reach the reporting threshold, the latency between two reports exclusively depends on the meeting pattern. Moreover, the users are ensured that their triplets are sufficiently jumbled before reporting. However, this strategy requires a high degree of trust in other users and introduces substantial overhead, which is not fairly distributed among the users. We observe extreme cases where users with thousands of triplets meet users with few triplets, causing a high overhead for the latter users. In comparison, the random-unfair exchange strategy requires a lower degree of trust and introduces less overhead. However, the partial exchanges of triplets diminishes the jumbling degree and can delay the reporting, as multiple meetings can be necessary to reach the report thresholds. The latency between two reports not only depends on the meeting pattern, but also on the random selection of the triplets. The overhead is also not equally distributed among the users. Finally, the random-fair strategy presents the most restrictive exchange condition, as the number of triplets to exchange is randomly determined and limited by the user with fewer triplets. The required degree of trust in other users is the lowest and there is no reporting overhead for any user. However, the restrictive exchange condition limits the amount of exchanged triplets that reduces the jumbling degree and can increase the latency between two reports, as the thresholds can be more difficult to reach.



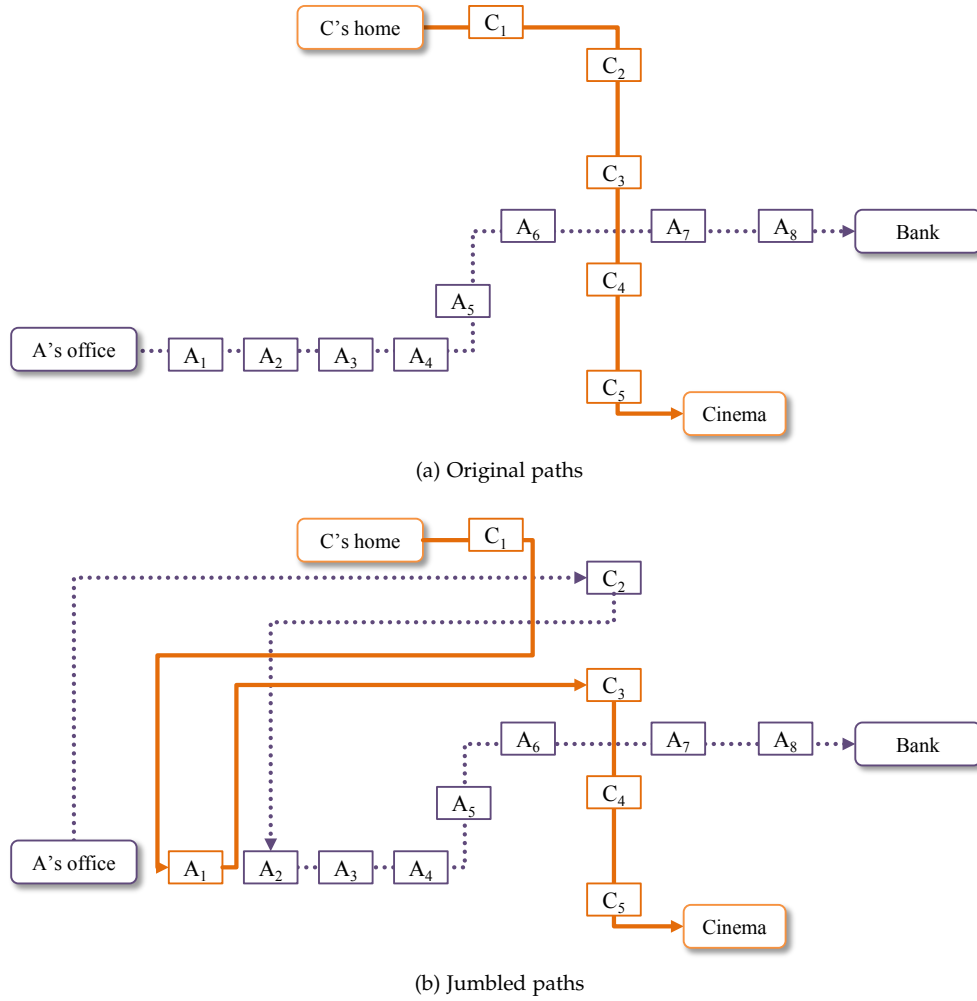


Figure 10: Examples of jumbled paths showing potential spatial outliers

#### 5.4 PRIVACY PROTECTION

We next quantify the privacy protection provided by our approach by means of extensive simulations. In our analysis, we model malicious application administrators who aim at distinguishing jumbled triplets from original ones based on their spatiotemporal information. In fact, jumbled triplets may have been collected in different geographical areas than the ones collected by the users themselves (see Figure 10a). The simultaneous transmission of both collected and exchanged triplets to the application server may hence generate paths showing spatial outliers as depicted in Figure 10b. We have implemented four different outlier detection algorithms and apply each of them on a realistic dataset in order to evaluate the performance of potential attackers in identifying jumbled triplets. We first present our simulation setup and method. In particular, we describe the main principles of the selected outlier detection algorithms and highlight their key parameters. Next, we describe our results and specially discuss the performance of these algorithms depending on the evaluated parameters.

##### 5.4.1 Simulation Setup and Method

Our evaluation is based on the GeoLife dataset introduced in Section 5.3.1. For our simulations, we selected a period of 14 days (from the 01/10/2008 to the 14/10/2008). In this period, we selected 20 users having at least met one other user and observed their mobility and meeting pattern. Figure 11 depicts the daily number of meetings during this period, while Table 4



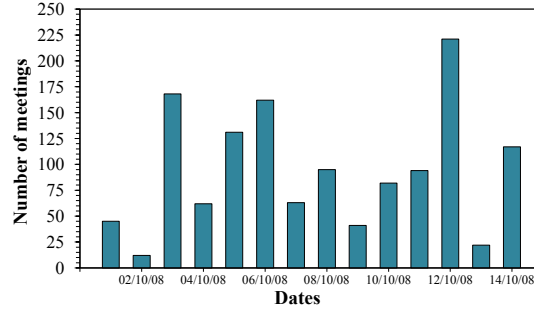


Figure 11: Distribution of daily number of meetings between the 01/10/2008 and the 14/10/2008

Table 4: User statistics ordered by number of meetings for the period of the 01/10/2008 to the 14/10/2008

USER ID	NUMBER OF MEETINGS	USER ID	NUMBER OF MEETINGS
117	4	86	78
126	5	42	78
12	8	11	103
151	9	83	108
142	15	13	111
122	18	52	112
102	21	82	112
123	31	68	204
62	46	94	222
124	59	10	235

details the number of meetings, trajectories, and collected triplets for each of the 20 selected users. Among these users, we further selected the following users who represent the extreme and average cases in terms of meetings: User 117 (four meetings), user 86 (78 meetings), and user 10 (235 meetings).

Next, we apply one of the following exchange strategies on the triplets collected by the users during the chosen period: (1) realistic with an exchange probability of 0.5, (2) random-unfair, and (3) random-fair (see Section 5.2.2). By doing so, we artificially generate jumbled paths. These paths are similar to those users would have obtained if they would have actually exchanged these triplets at physical encounters. After the jumbling, the triplets are chronologically ordered based on their timestamp, as if they would have been collected by the same user. We further assume that users report these triplets every hour in average.

#### *Detection of Jumbled Triplets*

In order to analyze the efficacy of the path hiding approach, we assume that malicious application administrators apply one of the following outlier detection algorithms tailored to identify spatial outliers at the end of the considered time period.

**VELOCITY-BASED OUTLIER DETECTION ALGORITHM** With this algorithm, we aim at identifying consecutive triplets that are inconsistent with a realistic movement model. In other words, our objective is to determine if the distance between two triplets can be covered within the period of time defined by their timestamps, i.e., if the user's velocity is realistic. To this end, we define different models based on a maximum velocity  $v_{\max}$  that cover most users' transportation modalities.

The algorithm computes the velocity  $v_i$  between consecutive triplets  $T_i$  and  $T_{i+1}$ . If the computed velocity is higher than  $v_{\max}$ , the triplet is considered as an outlier since its velocity

mismatches with the model. In our evaluation, we vary  $v_{\max}$  between one and 30 m/s in order to model pedestrians, cyclists, and vehicle passengers and drivers.

**GRUBB-BASED OUTLIER DETECTION ALGORITHM** Inspired from the Grubb’s test [SLZ03], this algorithm is based on the spatial distance between neighboring triplets and shown in Algorithm 1. In particular, it compares the Euclidian distance of each triplet  $T$  to the mean distance of its neighbors (see steps 3 to 12 in Algorithm 1). It hence differs from the above algorithm by the used metric and the number of considered triplets. In fact, it uses the distance (instead of the velocity) and considers both direct neighbors of each triplet as well as the neighbors of its direct neighbors (instead of a unique neighbor in the above algorithm). Note that additional neighbors can be taken into account in this algorithm. The computed spatial distance is then compared to the mean and the standard deviation obtained for all triplets by computing  $Z_s$  (see step 17). If  $Z_s$  is greater than the given confidence level  $c$ , the triplet is considered as outlier (see steps 18 and 19). The algorithm identifies an outlier per iteration and is hence iteratively applied when multiple outliers are assumed. As a result, the algorithm can be parametrized using the following parameters.

- ▷ Confidence level ( $c$ )
- ▷ Number of algorithm iterations ( $n$ )

In our evaluation, we investigate the impact of the selected confidence level and number of iterations on the performance of the algorithm in identifying jumbled triplets. Consequently, we have varied the number of interactions between one and 100. Recall that using 100 iterations allows the algorithm to identify 100 outliers. This approximately corresponds to assuming a jumbling degree of 50%, since the mean length of the considered paths is around 200 triplets. We have chosen values of 0.70, 0.95, and 0.99 for the desired confidence level. We have especially selected high values for the confidence level in order to only consider triplets identified as being exchanged with a high probability in order to reduce the number of false positives. Again, our analysis is by no means exhaustive, but aims at exploring the main characteristics of the algorithm.

**DISTANCE-BASED OUTLIER DETECTION ALGORITHM** This algorithm is based on the approach proposed in [KNT00]. In this approach, a triplet is considered to be a  $DB(p, d_{\min})$ -outlier if at least a fraction  $p$  of the triplets show a distance greater than the distance  $d_{\min}$  from the considered triplet [KNT00]. Let  $T_i$  be the triplet of interest. The algorithm, shown in Algorithm 2, first computes the Euclidian distance between  $T_i$  and each triplet of the same path (see step 3 in Algorithm 2). If the computed distance is lower than  $d_{\min}$ , the triplet is added in the neighborhood of  $T_i$  (see step 5). This means that the neighborhood of  $T_i$  contains the set of triplets that are within distance  $d_{\min}$  of  $T_i$ . According to [KNT00],  $p$  determines the minimum fraction of triplets that must be outside the neighborhood of an outlier. Consequently, the maximum number of triplets in the neighborhood of an outlier is equal to  $M = N(1 - p)$ ,  $N$  being the total number of triplets of the path. This implies that if there are at least  $M + 1$  triplets included in the neighborhood of  $T_i$ ,  $T_i$  is identified as a non-outlier and another triplet is examined (see steps 6 to 8). As compared to the previous algorithms, the distance-based outlier detection algorithm hence does not only compare the distance of each triplet to its direct neighbors (i.e., its successor and its predecessor), but the distance to any triplet until the newly defined neighborhood contains more than  $M + 1$  triplets. As a result, the algorithm can be parametrized according to the following values.

- ▷ Fraction of triplets ( $p$ )
- ▷ Distance ( $d_{\min}$ )

In our evaluation, we vary both parameters in order to explore the performance of the algorithm under various settings. We select the following values for  $p$ : 0.01, 0.05, 0.50, 0.75, and

**Algorithm 1** Grubb-based outlier detection algorithm**Input:**  $n$ : number of iterations,  $c$ : confidence level,  $N$ : total number of triplets

---

```

1: for each iteration  $i$  with  $i \in [1, n]$  do
2:   for all triplets  $T_j$  with  $j \in [2, N]$  do
3:     compute mean distance between  $T_{j-1}$  and its direct neighbors
4:      $\overline{d_{T_{j-1}}} = \frac{|T_{j-2}T_{j-1}| + |T_{j-1}T_j|}{2}$ 
5:     compute mean distance between  $T_j$  and its direct neighbors
6:      $\overline{d_{T_j}} = \frac{|T_{j-1}T_j| + |T_jT_{j+1}|}{2}$ 
7:     compute mean distance between  $T_{j+1}$  and its direct neighbors
8:      $\overline{d_{T_{j+1}}} = \frac{|T_jT_{j+1}| + |T_{j+1}T_{j+2}|}{2}$ 
9:     compute mean distance of the direct neighbors of  $T_j$ 
10:     $\overline{d_{T_{j-1}T_{j+1}}} = \frac{\overline{d_{T_{j-1}}} + \overline{d_{T_{j+1}}}}{2}$ 
11:    compute distance difference between  $T_j$  and its direct neighbors
12:     $s_{T_j} = |\overline{d_{T_{j-1}T_{j+1}}} - \overline{d_{T_j}}|$ 
13:  end for
14:  compute  $\overline{s_T}$  as the mean value of all  $s_{T_j}$  with  $j \in [2, N]$ 
15:  compute  $\overline{\sigma_T}$  as the standard deviation of all  $s_{T_j}$  with  $j \in [2, N]$ 
16:  for all triplets  $T_j$  with  $j \in [2, N]$  do
17:    compute  $Z_s(T_j) = \frac{s(T_j) - \overline{s_T}}{\overline{\sigma_T}}$ 
18:    if  $Z_s(T_j) > c$  then
19:       $T_j$  is categorized as outlier
20:      break
21:    end if
22:  end for
23: end for

```

---

1.00. Note that the lower  $p$ , the more outliers are assumed. Following the recommendations of [KNT00], we have chosen the following values for  $d_{\min}$  based on the distribution of the triplets: 1 m, 25 m, 50 m, 75 m, and 100 m.

**DENSITY-BASED OUTLIER DETECTION ALGORITHM** The last algorithm is adapted from [BKNS00]. It is based on the computation of a Local Outlier Factor (LOF) that quantifies the likeliness of a triplet to have been jumbled. Again, this method relies on the distance between triplets and determines the density of triplets around the triplet of interest. The density is defined as the number of nearest neighbors within a certain distance around a triplet. It hence measures how isolated a triplet is from potential neighbors. The rationale behind this algorithm is that outliers show lower density than non-outlier triplets. Algorithm 3 summarizes the different steps necessary to identify potential outliers. In particular, it requires the computation of the following variables introduced in [BKNS00] and adopted herein.

- ▷ The  $k$ -distance of a triplet  $T_i$  ( $k \in \mathbb{N}$ ) is defined in Algorithm 3 as the distance  $d_{T_i T_j}$  between  $T_i$  and a triplet  $T_j$  of the same path such that:

1. for at least  $k$  triplets  $T_k$  (with  $T_k \neq T_i$ ),

$$d_{T_i T_k} \leq d_{T_i T_j} \quad (1)$$

2. for at most  $k - 1$  triplets  $T_k$ ,

$$d_{T_i T_k} < d_{T_i T_j} \text{ with } T_k \neq T_i \quad (2)$$

- ▷ The  $k$ -distance neighborhood of a triplet  $T_i$  denoted  $N_k(T_i)$  contains every triplet whose distance from  $T_i$  is not greater than the  $k$ -distance of  $T_i$ . The triplets contained in the  $k$ -distance neighborhood of  $T_i$  are referred to as the  $k$ -nearest neighbors of  $T_i$ .

**Algorithm 2** Distance-based outlier detection algorithm**Input:**  $p$ : fraction of triplets,  $d_{\min}$ : distance

---

```

1: for all triplet  $T_i$  do
2:   for all triplet  $T_j$  with  $T_j \neq T_i$  do
3:     compute  $d_{T_i T_j} = |T_i T_j|$ 
4:     if  $d_{T_i T_j} \leq d_{\min}$  then
5:       add  $T_j$  in the neighborhood of  $T_i$ 
6:       if neighborhood size of  $T_i > M$  then
7:          $T_i$  is categorized as non-outlier
8:         break
9:       end if
10:    end if
11:  end for
12: end for

```

---

▷ The *reachability distance* of triplet  $T_i$  with respect to triplet  $T_j$  is defined as:

$$rd_k(T_i, T_j) = \max\{k\text{-distance}(T_j), d_{T_i T_j}\} \quad (3)$$

▷ The *local reachability density* of triplet  $T_i$  is defined as:

$$lrd_k(T_i) = \frac{|N_k(T_i)|}{\sum_{T_j \in N_k(T_i)} rd_k(T_i, T_j)} \quad (4)$$

▷ The *local outlier factor* of triplet  $T_i$  is defined as:

$$LOF_k(T_i) = \frac{\sum_{T_j \in N_k(T_i)} \frac{lrd_k(T_j)}{lrd_k(T_i)}}{|N_k(T_i)|} \quad (5)$$

As a result, the clustering of the triplets is based on both the minimum number of triplets  $k$  and the reachability distance  $rd_k$  [BKNSoo]. Both metrics define a density threshold denoted  $t_d$ . Triplets are considered as outliers when their LOF is below  $t_d$ , while they are considered as non-outliers for higher LOF values (see step 10 in Algorithm 3). In our evaluation, we therefore investigate the impact on the following parameters on the performance of malicious administrators in identifying jumbled triplets.

- ▷ minimum number of neighborhood triplets  $k$
- ▷ density threshold  $t_d$

We specially consider the following values for  $k$  based on the distribution of the triplets: 1, 25, 50, 75, and 100. For  $t_d$ , we examine the following values: 1, 5, 10, 15, and 20.

#### 5.4.2 Evaluation Metric

In the next step, we apply each of the presented outlier detection algorithms on the generated jumbled paths and vary their respective parameters. As a measure of the performance of the algorithm in identifying exchanged triplets, we use the Matthews Correlation Coefficient (MCC) computed as follows. True Positives (TP) are the number of exchanged triplets identified as exchanged ones. True Negatives (TN) are the number of original triplets identified as such. False Positives (FP) refer to the number of original triplets identified as exchanged, and False Negatives (FN) are the number of exchanged triplets identified as original.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \quad (6)$$

**Algorithm 3** Density-based outlier detection algorithm**Input:**  $k$ : minimum number of triplets,  $t_d$ : density threshold

---

```

1: for all triplets  $T_i$  do
2:   compute  $k$ -distance for  $T_i$  based on Eq. 1 and 2
3: end for
4: for all triplets  $T_i$  do
5:   determine the  $k$ -distance neighborhood of  $T_i$ 
6:   compute the reachability distance  $rd$  of  $T_i$  to all other triplets according to Eq. 3
7:   compute the local reachability density  $lrd$  of  $T_i$  according to Eq. 4
8:   compute the local outlier factor LOF according to Eq. 5
9:   if  $LOF \leq t_d$  then
10:     $T_i$  is categorized as outlier
11:   end if
12: end for

```

---

Table 5: Examples of MCC values for different TP and TN values (with  $TP = 1 - FP$  and  $TN = 1 - FN$ )

		TP				
		0.05	0.25	0.50	0.75	0.95
TN	0.05	-0.90	-0.71	-0.50	-0.28	0.00
	0.25	-0.71	-0.50	-0.25	0.00	0.28
	0.50	-0.50	-0.25	0.00	0.26	0.50
	0.75	-0.28	0.00	0.26	0.50	0.71
	0.95	0.00	0.28	0.50	0.71	0.90

Table 6: Chosen classification of the performance of the algorithms based on their MCC values

MCC VALUE	PERFORMANCE
$MCC \leq 0.00$	Poor
$0.00 < MCC < 0.30$	Fair
$0.30 \leq MCC < 0.60$	Good
$0.60 \leq MCC < 0.90$	Very good
$0.90 \leq MCC < 1.00$	Excellent
$MCC = 1.00$	Perfect

Table 5 illustrates examples of MCC values obtained for different values of TP and TN. The MCC values range between -1 and +1. A value of +1 indicates a perfect identification of both exchanged and original triplets, i.e., no false negatives or positives, while a value of -1 indicates a total failure in identifying both categories. In the following, the performance of the algorithms are evaluated based on the computed MCC values and categorized according to the classification presented in Table 6. For example, an algorithm is considered to perform *poorly* if its MCC value is negative or equal to zero. This corresponds to, e.g.,  $TP = 0.50$  and  $TN = 0.25$  (i.e., 50% of the exchanged triplets and 25% of the non-exchanged triplets have been correctly identified). We have repeated each simulation ten times and present the averaged results herein.

### 5.4.3 Results

In this section, we present and discuss the results obtained for each of the aforementioned outlier detection algorithms. We first analyze the MCC values obtained for 20 users of the dataset (see Section 5.4.1). Next, we examine the results of our three selected users (i.e., user 117, user 86, and user 10) in order to study the impact of the number of meetings on the performance of the algorithms in identifying jumbled triplets. For both scenarios, we specially analyze the influence of the applied exchange strategies on the performance of the chosen algorithms. Note that again, the obtained results depend on the characteristics of the utilized dataset. Our evaluation can therefore not be generalized to all possible scenarios, but again shows the feasibility of our approach under realistic assumptions.

#### Velocity-based Outlier Detection

Figure 12 illustrates the MCC values for different maximum velocity values ( $v_{\max}$ ). The figures in the left column depicts minimum, maximum, and quartiles of the MCC values computed for the 20 users of the dataset. The figures in the right column show the individual results for each of the three selected users. The first row of figures corresponds to the application of the realistic exchange strategy, while the second and third rows correspond to the random-unfair and random-fair strategies, respectively.

Figure 12a shows that the performance of the velocity-based outlier detection algorithm in identifying exchanged triplets when users apply the realistic exchange strategy is globally considered as *fair* to *good* according to our classification introduced in Table 6 (with the exception of extreme *poor* and *excellent* results). For  $v_{\max}$  equal to 20 and 30 m/s, the performance of the algorithm slightly degrades when considering the first quartile. In comparison, the results of the random-unfair exchange strategy illustrated in Figure 12c are globally better than those of the realistic strategy. Indeed, no first quartile is below zero, most medians are higher, and the third quartiles for  $v_{\max} = 20$  m/s and  $v_{\max} = 30$  m/s are greater than 0.60. Overall, the performance of the algorithm can hence be considered as *good* when users apply the random-unfair strategy. For the random-fair strategy, the performance of the algorithm further improves as shown in Figure 12e. Since all medians are greater than 0.30, the performance is categorized as *good* according to our classification. With the exception of  $v_{\max} = 5$  m/s, all first quartiles with the random-fair strategy are higher than those of the random-unfair strategy. Additionally, all third quartiles are included in the interval that corresponds to a *very good* performance. As a result, the realistic exchange strategy provides a better privacy protection since the algorithm performs worse in identifying jumbled triplets as compared to the other strategies. Note that the algorithm is, however, capable of identifying almost all jumbled triplets for a particular user having collected triplets in a remote location as compared to other users. This almost perfect identification is possible for a  $v_{\max}$  value of 10, 20, and 30 m/s independently of the applied exchange strategy due to the particular spatial distribution of the triplets.

With the exception of user 10, the results depicted in Figures 12b, 12d, and 12f shows that the performance of the algorithm improves with the value of  $v_{\max}$ . Concerning user 10, the performance globally drops with  $v_{\max}$  until reaching  $v_{\max} = 20$  m/s. This difference can be explained by the distribution of the velocity in his/her set of triplets. In fact, this user shows particularly low velocity between his/her own triplets and thus, a high  $v_{\max}$  allows the algorithm to better filter exchanged triplets. Again, applying the realistic exchange strategy provides globally a slightly better protection against curious administrators followed by the random-unfair and random-fair exchange strategies. For example, for  $v_{\max} = 30$  m/s and user 117,  $MCC = 0.51$  with the realistic strategy (see Figure 12b), whereas  $MCC = 0.80$  and  $MCC = 0.84$  for the random-unfair (see Figure 12d) and random-fair strategies (see Figure 12f), respectively. Moreover, these results show that globally the more meetings, the worse the performance of the algorithm to identify outliers and hence, the better protection against curious administrators. For example, for the random-unfair strategy, the performance of the algorithm in identifying the jumbled triplets of user 117 (four meetings) is considered

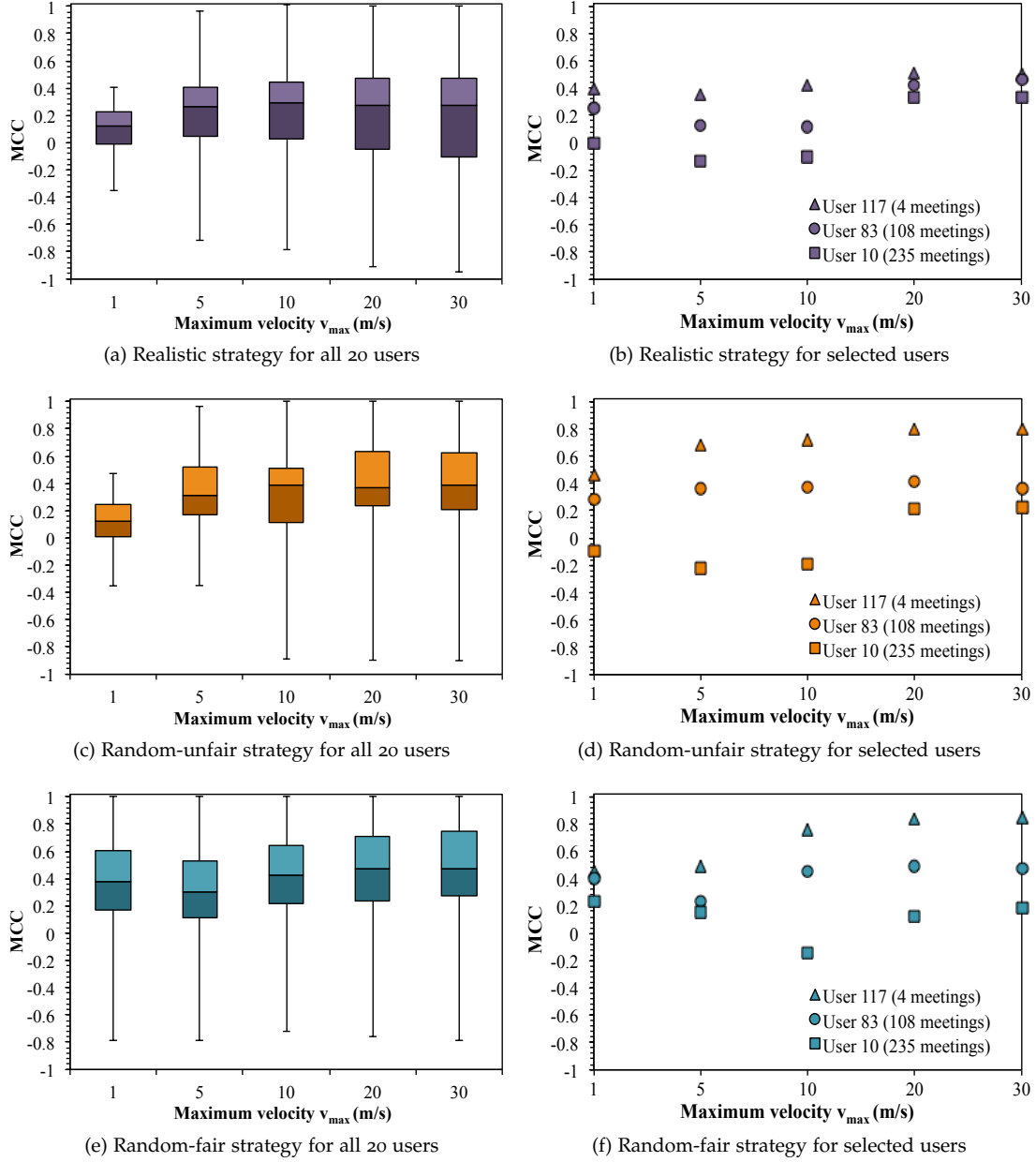


Figure 12: MCC for the velocity-based outlier detection algorithm with  $v_{\min} = 0$  and a uniform probability distribution

as *good* to *very good*, while it is considered as *good* for user 83 (108 meetings) and *poor* to *fair* for user 10 (235 meetings). However, the difference in the degree of protection depending on the number of meetings is less significant when the users apply the realistic exchange strategy and  $v_{\max}$  is greater than 20 m/s (see Figure 12b).

In summary, the velocity-based algorithm overall performs best in identifying exchanged triplets when clients apply the random-fair exchange strategy. In contrast, it performs worse when the realistic exchange strategy is applied. This means that the privacy of the users is better protected, when they apply the realistic strategy. Moreover, the results show that the fewer exchanges, the better identification of the exchanged triplets and vice versa.



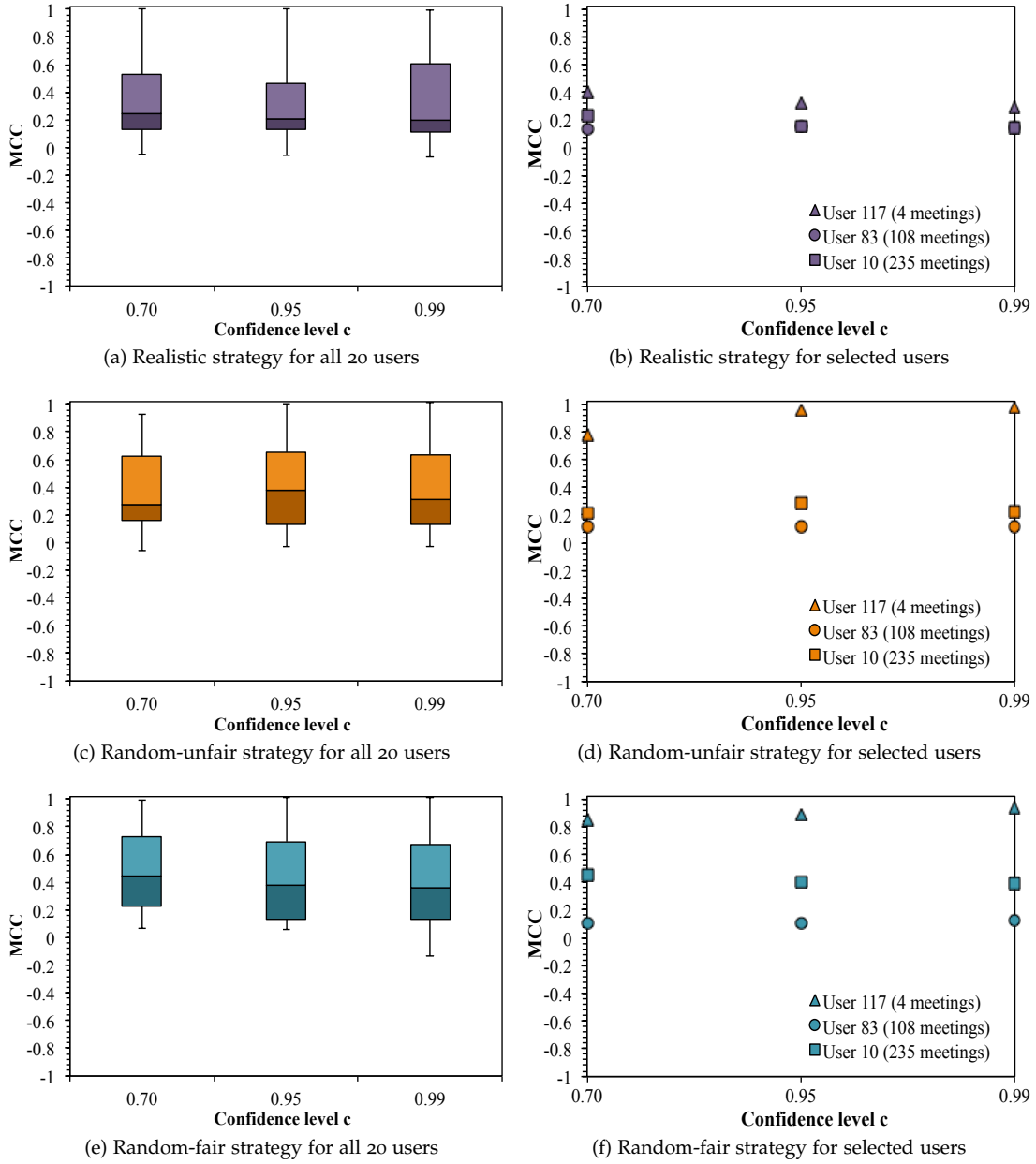


Figure 13: MCC for the Grubb-based outlier detection algorithm and different confidence levels  $c$  with  $n = 50$

#### Grubb-based Outlier Detection

When the Grubb-based outlier detection is applied, Figure 13 depicts the MCC values for different confidence levels ( $c$ ), while Figure 14 depicts them for different number of iterations ( $n$ ). Recall that the confidence level determines the threshold above which a triplet is considered as outlier based on the computed value of  $Z_s$ . The number of iterations determines how many times the algorithm is run and how many outliers are identified, as one outlier is identified per algorithm iteration.

According to the results illustrated in Figure 13, the performance of the Grubb-based algorithm in identifying jumbled triplets is considered as *fair* to *good* in most cases. For the random-unfair and random-fair strategies, the third quartiles are equal or greater than 0.60. The performance of the Grubb-based detection algorithm hence reaches the *very good* level. Moreover, the medians are higher for the random-fair strategy (cf. Figure 13e) followed by the



random-unfair strategy (cf. Figure 13c) as compared to the realistic strategy (cf. Figure 13a). Overall, this means that the algorithm performs the best when the random-fair exchange strategy is applied as compared to the random-unfair and realistic strategies.

For the selected users, the performance of the algorithm is best for user 117 independently of the applied exchange strategy, reaching even *excellent* results for the random-unfair and random-fair strategies as shown in Figure 13d and Figure 13f, respectively. This means that users with few meetings allow the algorithm to better identify jumbled triplets. However, in contrast to the previous algorithm, the performance does not significantly degrade with the number of meetings. In fact, the MCC values of user 10 (235 meetings) are greater than those of user 83 (108 meetings). Therefore, additional studies are necessary to determine the meeting threshold from which the performance degrades and also the dependency of the results with the distribution of the triplets. Similar trends as for the previous algorithm can be observed about the range of the algorithm's performance based on the applied exchange strategy and the number of meetings. When users apply the realistic strategy, the algorithm performs similarly for all users, while the difference in performance increases for the random-unfair and random-fair strategies based on the number of meetings. For example, for  $c = 0.95$ , the MCC values range between 0.16 and 0.33 for the realistic strategy (see Figure 13a), while they range between 0.10 and 0.90 for the random-fair strategy (see Figure 13e).

In Figure 14, we assume  $c = 0.95$  and vary the number of iterations  $n$ . Overall, the performance of the algorithm globally improves with the number of iterations for all exchange strategies until stabilization as shown in the figures in the left column. For the realistic strategy, the performance of the algorithm is considered as *poor* to *fair* for  $n = 1$ , while it is *fair* to *good* from  $n = 25$  (see Figure 14a). For the random-unfair and random-fair strategies, the performance is slightly better and is categorized as *fair* to *very good* based on the values of the quartiles as illustrated in Figure 14c and Figure 14e, respectively. However, the minimum MCC value is lower for  $n = 100$  than for  $n = 75$  for both the random-unfair and random-fair strategies. This means that the algorithm is about to reach the threshold above which the number of false positives increases. If all outliers have been already been identified, additional runs of the algorithm lead to the identification of original triplets as outliers and cause the decline in performance of the algorithm. The comparison of the medians shows that the algorithm performs globally better when users apply the random-fair exchange strategy compared to the random-unfair strategy. This means that the algorithm is able to better distinguish original triplets from jumbled ones. As a result, the random-fair strategy provides a lower privacy protection to the users as compared to the other strategies. In contrast, the realistic exchange strategy provides the best privacy protection as MCC values are lower.

For the selected users, the results confirm that the algorithm performs best when users had only few meetings. For example, user 117 (four meetings) shows a MCC value of up to 0.97 for the random-unfair strategy (see Figure 14d) and 0.90 for the random-fair strategy (see Figure 14f). This corresponds to *excellent* performance in our classification. Again, the influence of larger numbers of meetings on the performance need to be confirmed by additional studies, since the algorithm, e.g., performs worse for user 83 (108 meetings) than for user 117 (235 meetings).

In summary, the best privacy protection is again offered when users apply the realistic strategy and exchange their triplets multiple times. In contrast, the worst protection is provided with the combination of the random-fair strategy and rare exchanges with this detection algorithm.

#### Distance-based Outlier Detection

In this algorithm, a triplet is considered to be an outlier if at least a fraction  $p$  of the triplets show a distance greater than the distance  $d_{\min}$  from the considered triplet as previously introduced. We therefore study the impact of (1) the fraction of triplets  $p$  and (2) the distance  $d_{\min}$  on the performance of the algorithm in identifying jumbled triplets.

Let us first assume  $d_{\min} = 50$  and vary the fraction of triplets  $p$ . Figure 15 illustrates the corresponding results. For all exchange strategies, the variation of  $p$  does not influence the

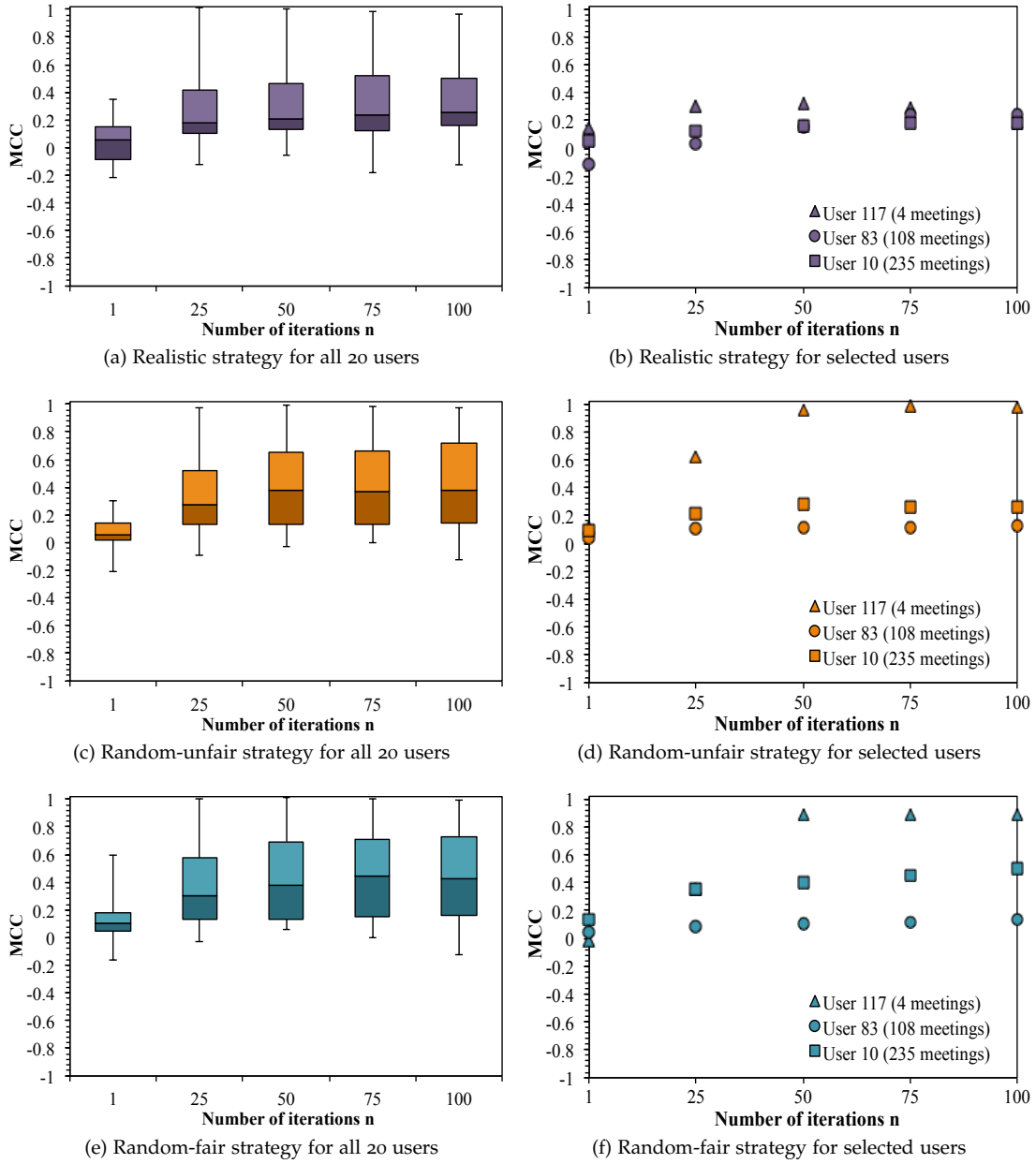


Figure 14: MCC for the Grubb-based outlier detection algorithm and different number of iterations  $n$  with  $c = 0.95$

performance of the algorithm. In fact, the MCC values remain almost the same for the 20 users of the dataset, including our three selected users as shown in Figures 15b, 15d, and 15f. As for both previous algorithms, the performance of the algorithm is slightly better for the random-fair strategy compared to the realistic and random-unfair strategies when comparing the quartiles. However, this algorithm globally performs worse than both the aforementioned velocity-based and Grubb-based outlier detection algorithms since all quartiles remain below 0.30, which corresponds to a *fair* performance only. In contrast to these algorithms, the number of meetings, however, does not impact the performance of the distance-based algorithm as depicted in Figures 15b, 15d, and 15f.

In the second step, we assume  $p = 50$  and vary  $d_{\min}$ . The results in Figure 16 highlight that the algorithm performs best for  $d_{\min} = 1$  for all exchange strategies. In this case, the performance of the algorithm is considered as *fair* to *very good* for all exchange strategies. For

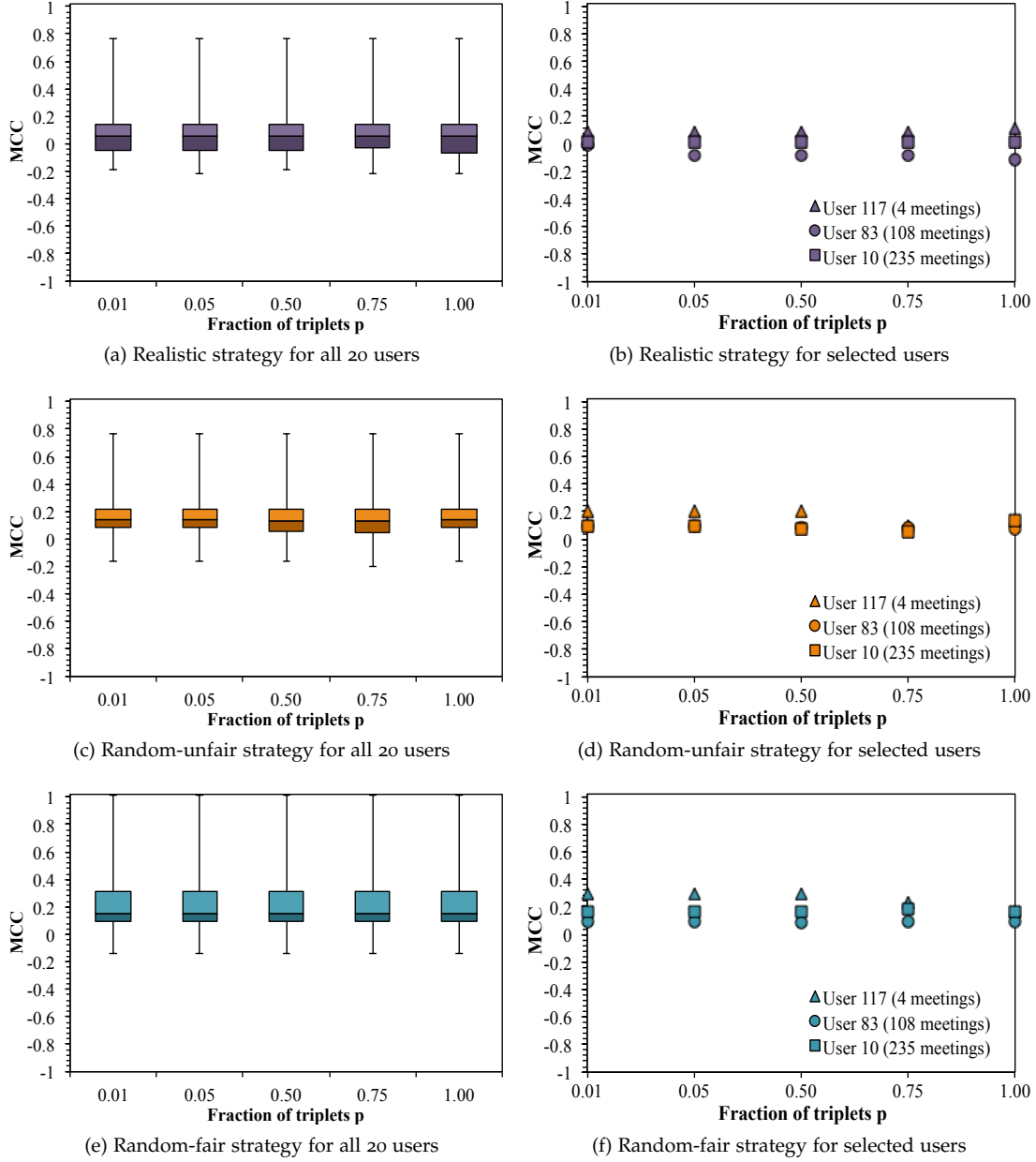


Figure 15: MCC for the distance-based outlier detection algorithm and different fractions of triplets  $p$  with  $d_{\min} = 50$

the remaining  $d_{\min}$  values, the performance is mainly *fair* with the exception of  $d_{\min} = 25$  for both the random-unfair and random-fair strategies. For these strategies, the third quartiles are greater than 0.30 and the performance is hence classified as *fair* to *good*. These results, however, highly depend on the spatial distribution of the dataset. Indeed, the best performance is reached for short distances, which represent the majority of triplets collected in nearby areas. The larger the distance, the worse the performance, as only few triplets have been collected in more remote locations. Therefore, these results cannot be easily generalized to other datasets. Again, the realistic strategy provides overall the best protection against malicious administrators, seconded by the random-unfair and random-fair strategies when comparing the quartiles. The observations about the performance of the algorithm for both  $d_{\min}$  and the applied exchange strategies are confirmed for the selected users as shown in

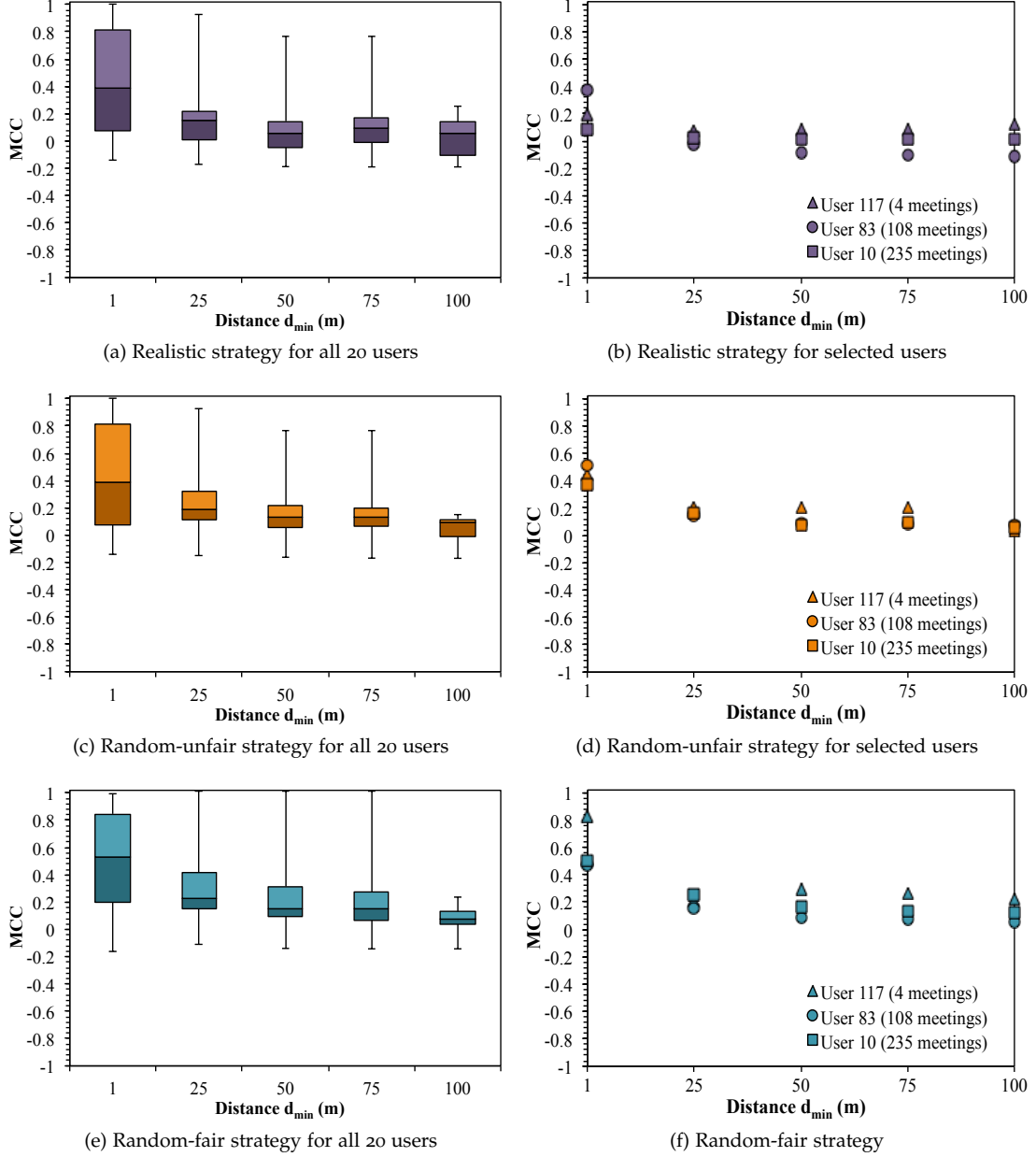


Figure 16: MCC for the distance-based outlier detection algorithm and different distances  $d_{\min}$  with  $p = 50$

Figures 16b, 16d, and 16f. Also, the number of meetings does not significantly influence the performance of the algorithm for different values of  $d_{\min}$ .

In summary, the results show that the distance-based outlier detection algorithm performs worse in identifying jumbled triplets when users apply the realistic exchange strategy and encounter many other users during their journeys.

#### Density-based Outlier Detection

We finally analyze the impact of the main parameters, i.e., the minimum number of triplets  $k$  and the density threshold  $t_d$ , of this algorithm on its performance in distinguishing jumbled triplets from original ones. Recall that  $k$  is the minimum number of triplets that are required to be in the neighborhood of the triplet of interest and  $t_d$  serves as threshold to determine whether a triplet is identified as outlier based on the computed Local Outlier Factor (LOF).

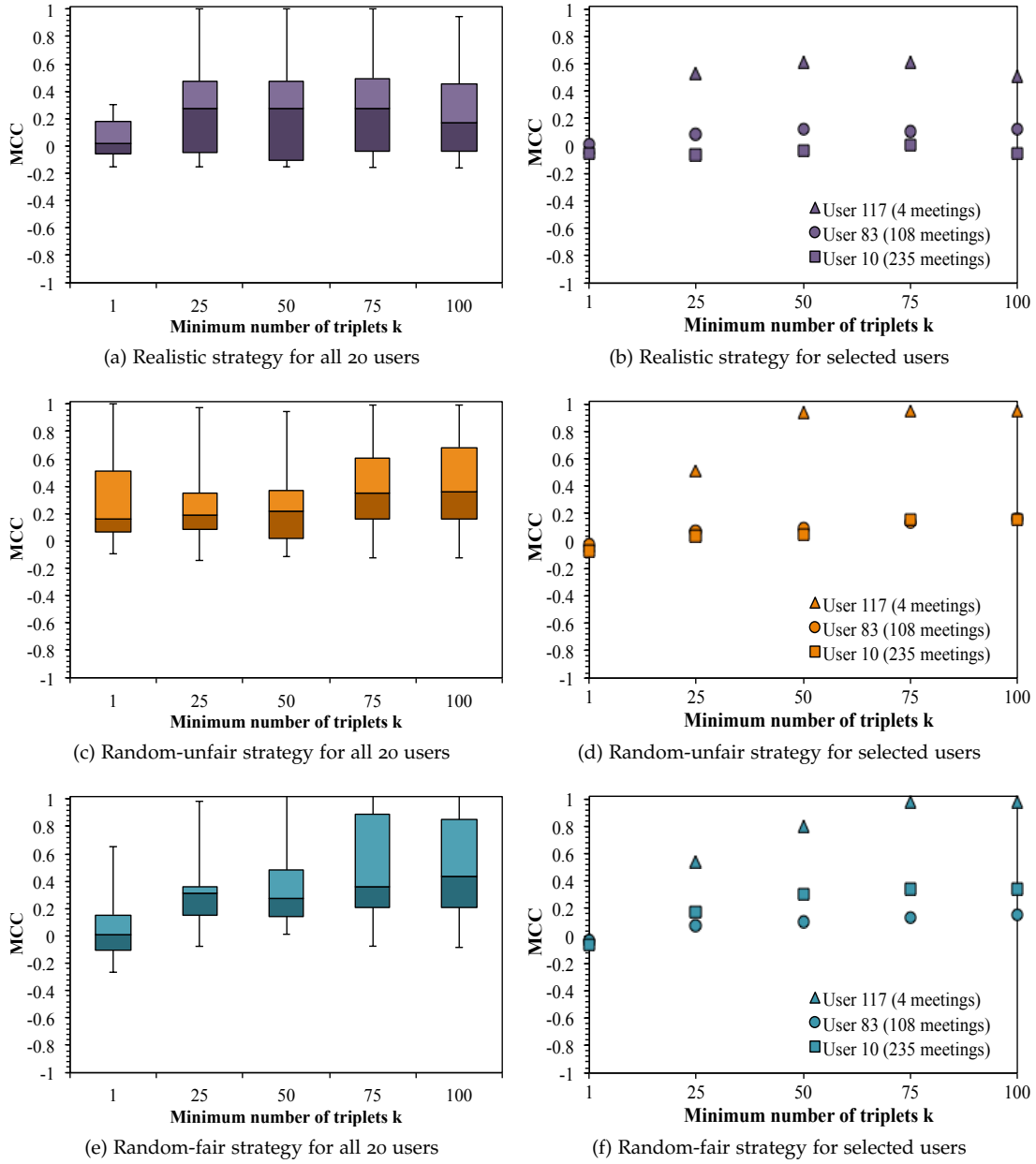


Figure 17: MCC for the density-based outlier detection algorithm and different minimum numbers of triplets  $k$  with  $t_d = 10$

Figure 17 presents the results for different values of  $k$  with  $t_d = 10$ , while Figure 18 presents those for different values of  $t_d$  with  $k = 50$ .

For the realistic strategy, the performance of the density-based outlier detection algorithm is considered as *poor* to *fair* for  $k = 1$ . It then improves to *good* from  $k = 25$  (see Figure 17a). In comparison, the performance is *fair* to *good* from  $k = 1$  to  $k = 50$  and *fair* to *very good* from  $k = 75$  for the random-unfair strategy (see Figure 17c). For the random-fair strategy, the algorithm shows *poor* to *fair* performance for  $k = 1$ , *fair* to *good* performance for  $k = 25$  and  $k = 50$ , and *fair* to *very good* performance from  $k = 75$  (see Figure 17e). Again, the algorithm shows the same behavior as the previous studied algorithms concerning the impact of the applied exchange strategies: The application of the random-fair strategy enables the algorithm to correctly identify more jumbled triplets than with the other strategies.

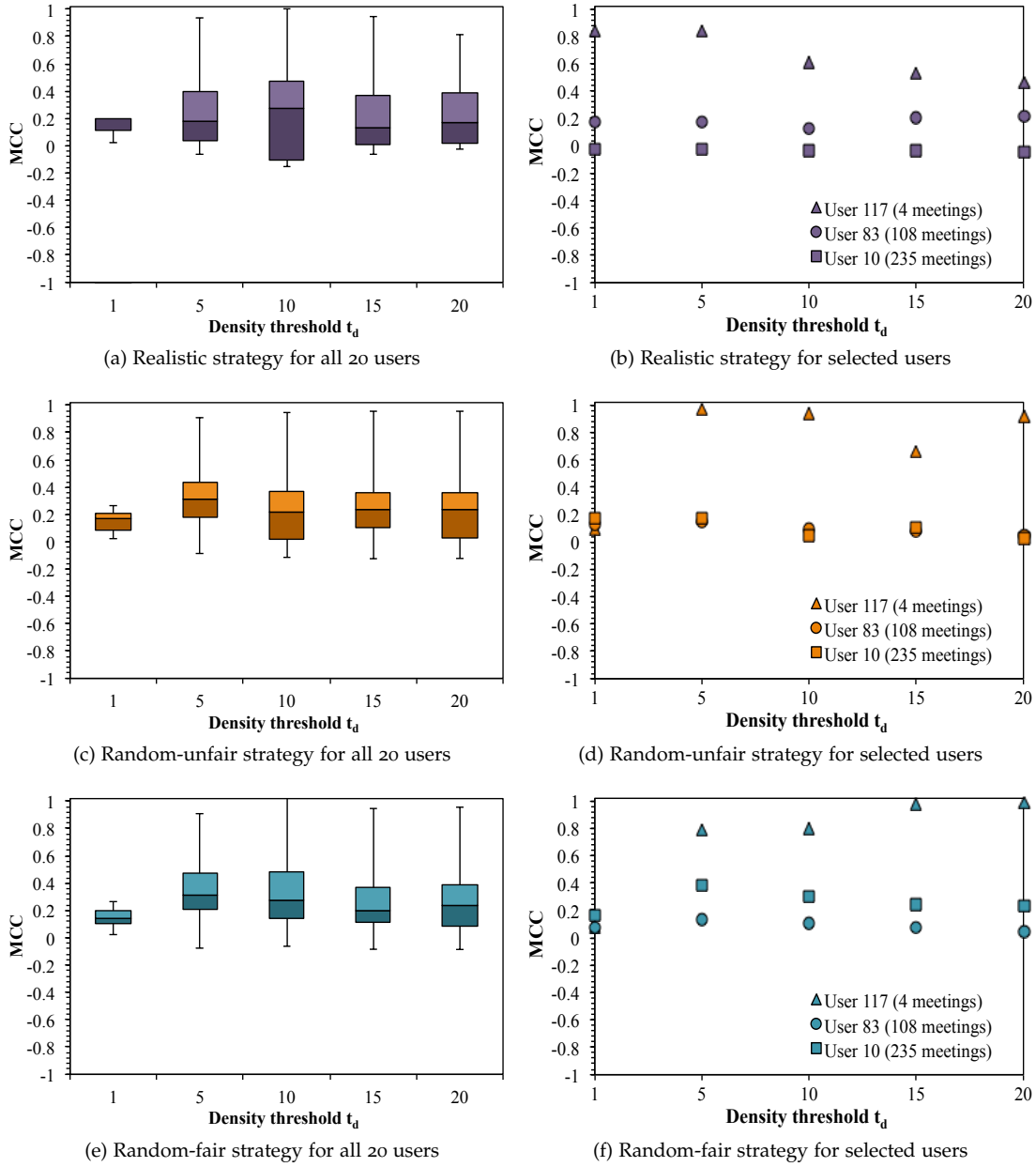


Figure 18: MCC for the density-based outlier detection algorithm and different density thresholds  $t_d$  with  $k = 50$

For the selected users, the same conclusions can be drawn. Moreover, few encounters lead to a better (correct) identification of jumbled triplets, while the difference between large numbers of encounters is insignificant. For user 117 (four meetings), the performance of the algorithm is *good* to *very good* when the user applies the realistic strategy (cf. Figure 17b), while its performance is *poor* to *fair* for the other users. For the random-unfair strategy (cf. Figure 17d), the performance for user 117 is up to *excellent*, whereas it is considered as *perfect* for the random-fair strategy (cf. Figure 17f). For the other users, the performance does not exceed *good* at any time.

Finally, we vary the density threshold  $t_d$  with  $k = 50$ . The performance of the algorithm in identifying exchanged triplets is considered as *fair* to *good* for all exchange strategies, with the exception of  $t_d = 10$  for the realistic strategy where first quartile is below zero (see Figure 18a). As previously determined, the random-fair strategy allows a better identification

of the jumbled triplets compared to the other strategies. Again, only few meetings lead to higher MCC values and potentially endanger the privacy of this particular user. For example, the algorithm shows *very good* to *perfect* performance for user 117 (four meetings) and all exchange strategies. For the other users, it remains *poor* to *fair* using the realistic strategy (see Figure 18b), *fair* using the random-unfair strategy (see Figure 18d), and *fair* to *good* using the random-fair strategy (see Figure 18f).

In summary, as for all studied outlier detection algorithms, the density-based algorithm performs worst when users apply the realistic exchange strategy in numerous exchanges. Therefore, the combination of the realistic strategy and a large number of meetings provides the better privacy protection for the users.

#### 5.4.4 Evaluation Summary

We have shown that the performance of the selected outlier detection algorithms in distinguishing jumbled triplets from original ones is globally not *excellent*, but remains *fair* to *good* with some *very good* exceptions. Indeed, the algorithms do not succeed in identifying all jumbled triplets with the exception of one particular user as shown by the low MCC values. They, however, perform better when users apply the random-fair strategy and have few meetings. In this case, only few triplets are exchanged at each encounter. They are hence more easily identifiable by the algorithm due to their potential difference in both time and space dimensions. By exchanging more triplets, the triplets may form groups that are coherent in terms of spatiotemporal distribution and prevent the algorithms from detecting them. We have, however, observed that some algorithms perform better for user 83 (108 meetings) than for user 10 (235 meetings) and vice versa. Therefore, additional studies are necessary to provide a more fine-granular analysis of the impact of large numbers of meetings on the performance of the algorithms. Based on the above results, the realistic exchange strategy has shown to provide the best privacy protection for the users, as all selected algorithms perform worst in identifying the jumbled triplets.

## 5.5 RELATED WORK

We have presented an approach to protect the privacy of the users by breaking the link between the spatiotemporal context of the sensor readings and the users who collect them. A simple alternative to our mechanism could be that the users use pseudonyms to report the triplets to the server (see Section 3.3.1). However, it was demonstrated in [Kru07], that the application can infer the real identity of the users by tracking their location traces over multiple reports, as it may expose the location of their workplaces and homes. In the current state-of-the-art detailed in Chapter 3, additional measures tailored to participatory sensing applications and preserving location privacy include spatial cloaking (cf. Section 3.3.2) and data perturbation (cf. Section 3.3.3). Spatial cloaking builds groups of users that share a common attribute (e.g.,  $k$  users located in the same district) to render them indistinguishable from each other. For example, the real location of the users can be replaced using the averaged location of the  $k$  nearest users [HKH10b]. However, spatial cloaking relies on a third-party entity managing the perturbation of the locations for all users. To generate the cloaked values, the users need to report their exact locations to the third-party entity, endangering their privacy. On the other hand, data perturbation intentionally perturbs the location traces by adding artificial noise (e.g., Gaussian noise) in order to conceal the individual traces. However, the noise model is selected by the application and influences the efficiency of the privacy protection. Indeed, independent random noise has been demonstrated insufficient to prevent adversaries from reconstructing the original data [Kru07]. The users must therefore trust the application to protect efficiently their privacy, while they ensure their privacy themselves in our approach.

Our approach shares features with the concept of mix zones [BS03], where the users change their pseudonyms when they encounter other users. However, our mechanism does not involve pseudonyms, but is solely based on the triplets collected by the users, which are



actually exchanged. Our concept shares also similarities with [HG05], [MC09] and [MC10]. In these schemes, the paths followed by users are replaced by newly generated paths intersecting with those of other users. However, these schemes are centrally organized and require users to transmit their current and actual position to a trusted entity for the generation of the new paths. Finally, our scheme shares similarities with the data aggregation scheme proposed in [SZLZ10]. This decentralized scheme is based on data slices equally distributed between neighbors before being reported to an aggregation server. Instead of only considering an equal distribution between neighbors, we examine multiple exchange strategies as well as reporting strategies that are not addressed in this prior work. Moreover, we investigate different dimensions in our evaluation and we do not assume the existence of an aggregation server. Our mechanism is tailored for common participatory sensing applications, where each user individually and directly reports triplets to the application server.

## 5.6 SUMMARY

In this chapter, we have presented a collaborative and decentralized approach to preserve the location privacy of users contributing to participatory sensing applications. Our approach is solely based on the exchange of the collected sensor readings between users in physical proximity in order to conceal the paths they have followed. We have first examined multiple exchange strategies in combination with different reporting strategies to determine their impact on the jumbling degree, the distance, and the incurred reporting overhead. Based on the utilized real-world dataset, we have shown that the realistic exchange strategy guarantees privacy independently of the applied reporting strategy. However, its application requires a high degree of trust in other users and generates a potentially large reporting overhead for some of the users. In comparison, the random-unfair and random-fair strategies require a larger amount of meetings to ensure similar guarantees, but require a lower degree of trust and generate less overhead. In the second step, we have used state-of-the-art outlier detection techniques that we have tailored to the application domain in order to investigate if exchanged triplets can be distinguished. The results show that the performance of the mechanisms in distinguishing exchanged triples from original ones is mostly between *fair* and *very good* on a scale ranging from *poor* to *perfect*. This means that only partial identification of jumbled triplets is possible in most cases. In particular, the realistic exchange strategy globally provides the best privacy protection followed by the random-unfair and random-fair strategies. As a result of our evaluation, we have provided insights for the strategy selection depending on the desired privacy guarantees, the degree of trust in other users, and the willingness to equally share the reporting overhead. Using our approach, the users can therefore customize their exchange and reporting settings depending on their preferences and, protect their location privacy themselves.



## TRUSTMETER: A TRUST ASSESSMENT SCHEME FOR COLLABORATIVE PATH HIDING

---

The best way to find out if you can trust somebody is to trust them.

---

E. Hemingway

In our collaborative path hiding mechanism detailed in Chapter 5, users mutually preserve their privacy in a decentralized fashion by exchanging sensor readings at physical encounters. Swapping a subset of the data samples removes the association between the sensor readings and the identity of the users who collected them. The sensor readings hence do not reveal the actual paths followed by each user, but instead result in jumbled paths which cannot be easily reconstructed. This scheme, however, relies on the collaboration of the participating users. Malicious users may attempt to disturb the operation of the mechanism by, e.g., dropping exchanged data samples as outlined in Section 5.1.2. This can lead to a total failure of the mechanism and thus of the underlying application. To overcome these threats, we propose an application-agnostic scheme called TrustMeter that assesses the behavior of the users contributing to the collaborative path hiding process [CRPSKH12a].

In Section 6.1, we first recall our assumptions and models already adopted in Chapter 5, before describing our novel scheme in Section 6.2. We discuss the robustness of TrustMeter against threats to reputation in Section 6.3 and assess its performance in identifying malicious users in Section 6.4. We further quantify the traffic overhead incurred by TrustMeter in Section 6.5. Finally, we survey related work in Section 6.6. Section 6.7 concludes this chapter.

### 6.1 SYSTEM AND THREAT MODELS

In this section, we provide an overview of our system and threat models. As we adopt the same models as in the previous chapter, further details are also provided in Sections 5.1.1 and 5.1.2, respectively. In particular, we refine the adversary model adopted in this chapter.

#### 6.1.1 System Model

We assume that mobile devices automatically collect sensor readings. Each sensor reading  $s$  is stamped with the collection time  $t$  and location information  $l$  forming a triplet  $\langle t, l, s \rangle$ . Instead of directly reporting the triplets to the application server, the mobile devices leverage the collaborative path hiding concept introduced in Section 5.2. This means that mobile devices physically exchange triplets at opportunistic encounters according to one of the following exchange strategies detailed in Section 5.2.2.

- ▷ **REALISTIC STRATEGY** Users exchange the entire set of collected and/or exchanged triplets with a certain probability at encounters.
- ▷ **RANDOM-UNFAIR STRATEGY** Each user independently and randomly determines the number of triplets to exchange.
- ▷ **RANDOM-FAIR STRATEGY** Users agree on a common number of triplets to exchange at each meeting.

Each user uploads a combination of the samples received from encountered users and the remainder of their own collected samples to the application server every hour as assumed in Chapter 5.

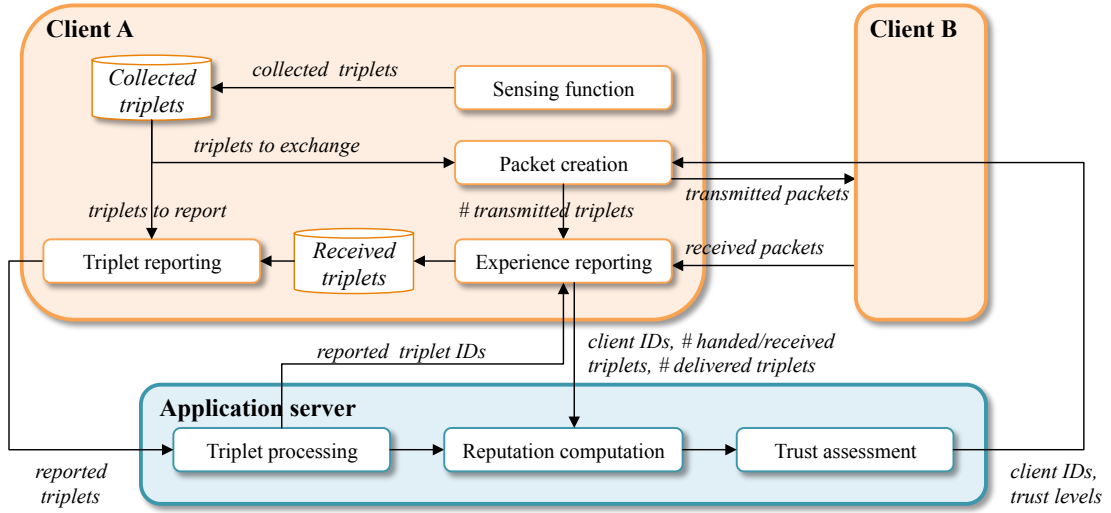


Figure 19: Overview of the TrustMeter architecture

### 6.1.2 Threat Model

In our threat model detailed in Section 5.1.1, we assume two kinds of adversaries: (1) honest-but-curious application administrators and (2) malicious users. In this chapter, we specially consider malicious users, who either drop exchanged triplets or create false triplets to exchange. By conducting such attack, malicious users aim at altering the results consolidated at the server side, and perturbing the function of the collaborative path hiding mechanism. In the following, we refer to malicious users who neither exchange nor report triplets from other users as *droppers* and to users who create falsified triplets as *spammers*.

As users apply the collaborative path hiding mechanism, application administrators do not obtain complete information about the paths followed by the users during the sensing process as shown in Chapter 5. In order to breach the privacy of the users, administrators may, however, be interested in any information about exchanges reported to the server that can help them in linking the jumbled triplets to the users who gathered them.

## 6.2 TRUSTMETER ARCHITECTURE

TrustMeter is built upon our system model presented in Section 6.1.1. It comprises clients (i.e., mobile devices) and an application server as illustrated in Figure 19. The objective of TrustMeter is two-fold: (1) identify malicious clients that launch dropping and spamming attacks as defined in Section 6.1.2, and (2) quarantine malicious clients in order to mitigate their attacks. We first provide an overview of the system architecture, before detailing the underlying mechanisms.

### 6.2.1 Overview

We assume that clients have a unique identifier  $ID_{client}$  and the application server has a public/private key pair. Each client periodically transmits the triplets to the application server. At each transmission, the application server disseminates a table containing the level of trust of all users. The level of trust is computed by the server based on the triplets reported by all users and their feedback about past exchanges. The feedback is reported to the server with the transmitted triplets in form of statistics.

When a client encounters another client, the two entities first examine the trust level of the opposite party provided by the application server. Only if both clients are rated as *trusted*, they

initiate the exchange of triplets using one of the three strategies (cf. Section 6.1.1). They first generate a unique identifier for each triplet and encrypt it using the public key of the server in order to prevent other clients from accessing its content, before physically exchanging them.

### 6.2.2 Underlying Mechanisms

Next, we present the building blocks of TrustMeter and detail their integration into a common participatory sensing application.

#### *Preparation and Exchange of Triplets*

When two clients meet, they first examine the trust level of the opposite party and only initiate the exchange of triplets if none of them is rated as *untrusted*. The triplets to be exchanged are individually encrypted using the public key of the server to prevent other users from accessing their content. For each triplet  $i$ , clients generate a unique identifier ( $ID_i$ ) by concatenating their own identifier ( $ID_{client}$ ), the time of creation of the triplet ( $t_i$ ), and a random number ( $r$ ), and computing a one-way hash function of the result.

$$ID_i = H(ID_{client} \parallel t_i \parallel r) \quad (7)$$

The use of concatenation and hashing results in the creation of unique triplet Identifiers (IDs) that cannot be easily guessed by potential attackers. Note that other mechanisms for the generation of unique triplet identifiers can be used without loss of generality.

Before sending triplets to another client, each client stores the IDs of the triplets to be exchanged and the identifier of its exchange partner  $ID_{partner}$ . Both clients exchange their triplets. Additionally, they set a delivery timeout  $T_{delivery}$  specific to the exchanged triplets. If these triplets do not reach the server before the timeout expiration, the clients consider them as lost and exchange them again.

#### *Reporting of Triplets*

When reporting new triplets to the server, clients can (1) verify if triplets they have exchanged with other clients have been delivered to the server and (2) report their experience with other clients, as detailed in the following.

**TRACKING OF TRIPLETS** The clients request the list of the triplet IDs delivered to the server since their last connection. Based on this list, the clients determine how many exchanged triplets have been successfully delivered to the application server. Note that clients cannot directly request the IDs of the triplets of interest to the server, since doing so would allow the server to establish a link between the triplet ID and previously uploaded reports and thus, defeat the primary purpose of the collaborative path jumbling mechanism. If the transmitted triplets have reached the server, the clients delete them. If not, they wait until the expiration of the timeout to retransmit them.

**EXPERIENCE REPORTING** The clients report their experience with all exchange partners by compiling statistics for each pair-wise exchange. These statistics include the identifiers of the two entities involved, the number of triplet exchanges in both directions, and the number of triplets delivered to the server. By reporting these statistics, the clients provide insights about the exchange pattern to the server in order to improve the reliability of the feedback provided by the clients and prevent malicious users from fraudulently increasing their reputation. However, the server cannot link the identity of the clients and the triplets they generated based on these statistics, as neither the IDs of the exchanged triplets nor the fraction of own collected triplets are disclosed.

### Reputation Computation and Trust Assessment

We describe herein how the server computes the reputation scores and derives the associated trust level for each client. We introduce two reputation scores,  $R$  and  $R'$ .  $R$  measures the participation of the clients in the collaborative path hiding process based on the numbers of exchanged triplets and delivered triplets. As a result, the value of  $R$  provides insights about potential dropping behavior. In contrast,  $R'$  focuses on identifying potential spamming behavior based on the reported triplets. For each new client, the server initializes  $R$  with  $R_0$  and  $R'$  with  $R'_0$  in order to allow new clients to participate in exchanges with already active clients.

$R$  is recursively computed according to Equation 8 each time clients report statistics. Equation 8 corresponds to an exponential moving average with a coefficient of 0.25.  $R \in [0,100]$  and  $\delta$  is the ratio of the number of delivered triplets to the total number of exchanged triplets. The lower the value of  $\delta$ , the higher the probability that the client has dropped packets.  $R$  increases for  $\delta > \frac{R_n}{100}$ , while it decreases for lower values.

$$R_{n+1} = R_n + \frac{100 \cdot \delta - R_n}{4} \quad (8)$$

For the computation of  $R'$ , we assume that the server can detect falsified triplets referred to as spam, using a function such as applied in [HKH10a]. When the server identifies spam, it flags the client having reported these triplets as well as its exchange partners. The server computes two metrics as indicator of spamming behavior,  $\gamma_1$  and  $\gamma_2$ .  $\gamma_1$  is the ratio of the number of flagged exchanges over the total number of exchanges of the client.  $\gamma_2$  is the ratio of the number of reported triplets identified as spam over the total number of reported triplets. The higher  $\gamma_1$  and  $\gamma_2$ , the higher the probability that the clients have injected spam. Based on these metrics, the server computes  $R^{(1)}, R^{(2)}, R^{(3)} \in [0,100]$  as follows, with  $\gamma = \frac{\gamma_1 + \gamma_2}{2}$ .

$$R_{n+1}^{(1)} = R_n^{(1)} + \frac{100 \cdot (1 - \gamma_1) - R_n^{(1)}}{4} \quad (9)$$

$$R_{n+1}^{(2)} = R_n^{(2)} + \frac{100 \cdot (1 - \gamma_2) - R_n^{(2)}}{4} \quad (10)$$

$$R_{n+1}^{(3)} = R_n^{(3)} + \frac{100 \cdot (1 - \gamma) - R_n^{(3)}}{4} \quad (11)$$

Next, the server computes  $R'$  according to Algorithm 4, with  $\lambda_U$  and  $\lambda_T$  being the reputation thresholds used by the server to classify the clients into the categories introduced in Table 7. The rationale behind this algorithm is to first identify clients that are clearly malicious (i.e.,  $R^{(3)} < \lambda_U$ ) or honest (i.e.,  $R^{(3)} > \lambda_T$ ). Then, the classification is refined for clients that are still considered as *indefinite*. Step 1 in Algorithm 4 focuses on identifying additional honest clients based on the value of  $R^{(2)}$ , while potential malicious clients remain classified as *indefinite*. Similarly, step 3 focuses on identifying malicious clients based on the value of  $R^{(1)}$ . Finally, the server selects the minimum of  $R_{n+1}$  and  $R'_{n+1}$  as reputation value and associates the corresponding trust level to the client according to the mapping presented in Table 7. Choosing the minimum prevents malicious clients from masking spamming behavior by well behaving in terms of dropping and vice versa. Note that we chose  $R_0$  and  $R'_0$  as the average of  $\lambda_U$  and  $\lambda_T$  to allow new clients to take part in exchanges without introducing unjustified promotion or penalty.

### Dissemination of Trust Levels

When a client reports triplets to the server, it obtains an update of the trust levels of all other clients. These levels are piggybacked with the acknowledgements from the server in order to

**Algorithm 4** Computation of  $R'$ 


---

**Input:**  $R^{(1)}, R^{(2)}, R^{(3)}$

```

1: if  $\lambda_U \leq R^{(3)} \leq \lambda_T$  and  $R^{(2)} > \lambda_T$  then
2:    $R' \leftarrow R^{(2)}$ 
3: else if  $(\lambda_U \leq R^{(3)} \leq \lambda_T$  and  $R^{(1)} < \lambda_U)$  then
4:    $R' \leftarrow R^{(1)}$ 
5: else
6:    $R' \leftarrow R^{(3)}$ 
7: end if
8: return  $R'$ 

```

---

Table 7: Mapping of trust levels with reputation values

TRUST LEVEL	REPUTATION RANGE
Untrusted	$0 \leq \min(R, R') < \lambda_U$
Indefinite	$\lambda_U \leq \min(R, R') \leq \lambda_T$
Trusted	$\lambda_T < \min(R, R') \leq 100$

reduce the incurred traffic and avoid that the server obtains temporal information about the exchanges. Clients leverage these levels to decide to exchange triplets with newly encountered clients as introduced in Section 6.2.2.

## 6.3 ROBUSTNESS AGAINST THREATS TO REPUTATION AND PRIVACY

We assume the threat model of Section 6.1.2 and discuss the resilience of TrustMeter against the following attacks.

## 6.3.1 Manipulation of Reputation

Malicious clients can attempt to alter the reputation computed and stored at the server side. The application server is, however, protected against unauthorized access using standard cryptographic primitives. As such, this particular attack cannot be launched. Furthermore, malicious clients can try to report falsified information on behalf of others. Our scheme protects honest clients against this attack by requesting clients to authenticate with the application server. Such an attack would hence only be successful if malicious clients can access the private keys of the targeted clients, which is beyond the scope of our attacker model. Malicious clients can send falsified information to the server about their exchanges in order to: (1) reduce the reputation of other clients by indicating that they have got fewer triplets from other clients and/or fewer triplets were delivered, or (2) fraudulently increase their own reputation by reporting larger numbers of handed and delivered triplets. However, recall that both exchange partners provide the numbers of handed, received, and delivered triplets to the server. This means that the server can verify that the provided information is the same for both exchange partners. If the figures do not match, the server assumes that one of the exchange partners is providing false information. It then observes the behavior of both exchange partners by maintaining a counter of non-matching reports in order to distinguish lying from honest clients. The larger the counter value, the higher the probability that a client is malicious. Another more sophisticated attack is the collusion of malicious clients that may agree on corroborating falsified information to mutually increase their reputation. Nevertheless, the efficacy of this attack increases with the frequency at which it is conducted. Simultaneously, exchanges between the same clients at a abnormal high frequency increases the risk to be detected by the server, which maintains a list of exchange partners for each client. As a result, the impact of such attacks remains limited. In order to still diminish the effect of this

attack, the server can take into account the number of exchange partners in the reputation computation. The more exchange partners, the higher the increase in reputation. At the same time, honest clients showing a small number of partners due to their physical distribution should not be penalized.

### 6.3.2 *Replay and Sybil Attacks*

Under normal operating conditions, triplet retransmissions are permitted after expiration of the delivery timeout. However, malicious clients can abuse this feature in order to increase their reputation by replaying triplets to exchange and/or to report. The efficacy of this attack depends on the frequency at which it is launched. The server receiving the same triplets at abnormal rates considers them as spam and utilizes the mechanisms described in Section 6.2.2 to identify malicious clients. Moreover, TrustMeter does not prevent malicious clients from adopting multiple identities. However, malicious clients need first to gain reputation before effectively launching attacks, since the trust level of new clients is initialized as *indefinite*. This thus increases the cost of such attack for malicious clients.

In summary, we have shown that TrustMeter can identify a variety of threats against the reputation system by malicious clients.

## 6.4 IDENTIFICATION OF MALICIOUS USERS

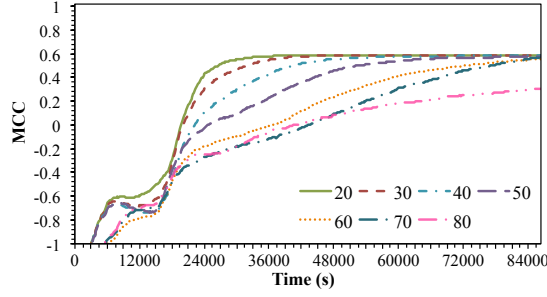
We analyze the performance of TrustMeter in identifying malicious behaviors (i.e., spamming and dropping) by means of extensive simulations. In particular, we examine the impact of the proposed exchange strategies as well as different framework parameters.

### 6.4.1 *Simulation Setup and Method*

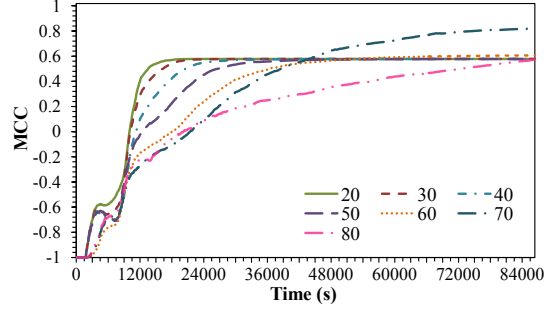
We implemented both our system model (see Section 6.1.1) and the TrustMeter scheme (see Section 6.2) in the ONE simulator [KOK09]. In the following, we consider a constant population of 100 clients over 24 hours for each simulation. The clients follow a pedestrian movement model based on a section of Helsinki City. All clients contribute to a participatory sensing application by generating a new triplet every 150 s and reporting triplets to the server every hour. The collected sensor modalities are, however, irrelevant since TrustMeter is application-agnostic. We assume that the client devices use Bluetooth to communicate with neighboring devices. We also assume lossless communication within a radius of 10 m, which is consistent with typical Bluetooth communication range. In order to study the impact of the exchange strategies introduced in Section 6.1.1, we assume that clients exchange triplets with clients rated as *indefinite* or *trusted* using one of the following strategies: (1) realistic, (2) random-unfair, and (3) random-fair. Moreover, they do not exchange triplets with *untrusted* clients. Depending on the simulation scenario, each client can act as an honest client, a dropper, or a spammer.

As a measure of the TrustMeter performance in identifying malicious clients, we use the Matthews Correlation Coefficient (MCC) introduced in Equation 6 (cf. Section 5.4.1). In this case, True Positives (TP) are the number of malicious clients identified as malicious. True Negatives (TN) describe the number of honest clients identified as such. False Positives (FP) are the number of honest clients identified as malicious, and False Negatives (FN) are the number of malicious clients identified as clients. Moreover, we consider malicious clients and honest clients that are both classified as *indefinite* as false negatives and false positives, respectively.

Recall that a MCC value of +1 indicates a perfect identification of both malicious and honest clients, i.e., no false negatives or positives, while a value of -1 indicates a total failure in identifying both categories. In the following, we use the classification introduced in Table 6 in order to analyze the performance of TrustMeter based on the obtained MCC values.



(a) Realistic strategy with an exchange probability of 0.5



(b) Realistic strategy with an exchange probability of 1.0

Figure 20: Impact of the exchange probability in the realistic strategy on the MCC for different values of the initial reputation scores  $R_0$  and  $R'_0$  (with  $R_0 = R'_0$ )

Additionally, we measure the impact of the identification of malicious clients on the latency until the triplets reach the server. We repeated each simulation 50 times using different random seeds to model different movement patterns. We present herein the averaged results.

#### 6.4.2 Framework Parameterization

We first examine the performance of TrustMeter in identifying malicious clients depending on the following parameters: (1) initial reputation scores, (2) reputation thresholds, and (3) delivery timeout. For these simulations, we consider a constant population of 90 honest clients, five droppers (dropping all packets), and five spammers (spamming all packets).

##### Initial Reputation Scores ( $R_0$ and $R'_0$ )

We first investigate the impact of  $R_0$  and  $R'_0$  on the performance of TrustMeter. Recall that  $R_0$  and  $R'_0$  are the initial reputation scores attributed by the server to new clients in order to allow them to exchange triplets with already active clients. As detailed in Section 6.2.2, we select  $R_0 = R'_0$ , with  $R_0$  and  $R'_0 \in [0, 100]$ . Moreover, we assume  $|\lambda_U - \lambda_T| = 20$  for these simulations. Moreover, the exchange probability for the realistic strategy is chosen to be 1 in order to compare the exchange strategies on the same basis, i.e., the number of exchanges. If the exchange probability is set to 0.5 in order to prevent the application server from reconstructing the original paths of the users, the obtained MCC values are similar to those obtained when the users exchange triplets at each encounter, but it takes around twice as much time to reach the same MCC values as shown in Figure 20.

Figure 21 illustrates the evolution of the MCC for different initial reputation scores over time. Figure 21a presents the results obtained when all clients apply the realistic strategy, while Figure 21b and Figure 21c present the same for the random-unfair and random-fair strategies, respectively. The results show an initial bootstrapping phase in which the MCC values fluctuate due to a low number of exchanged and reported triplets. During this phase, each triplet significantly impacts the reputation of the clients and the associated trust levels.



After this phase, the system stabilizes and attributes a definitive trust level to the client. If the behavior of the clients changes, TrustMeter adapts the attributed trust level to the current behavior of the clients as shown in Sections 6.4.3 and 6.4.4. This process, however, takes some time as the trust levels are derived from observations of the past behavior of the clients.

Figure 21 shows that high initial reputation scores globally lead to a better performance of TrustMeter in identifying malicious clients. For example, TrustMeter reaches *perfect* identification at  $t=65,000$  s for initial reputation scores of 70 and 80 when clients apply the random-fair exchange strategy (see Figure 21c). In comparison, TrustMeter performs worse for lower reputation scores. For initial reputation scores chosen between 20 and 50, the identification is however considered as *very good*, whereas it is *excellent* for  $R_0=R'_0$  at  $t=65,000$  s, for example. Since the reputation thresholds  $\lambda_U$  and  $\lambda_T$  are centered by design on  $R_0$  and  $R'_0$  (cf. Section 6.2.2), the lower the initial reputation scores, the lower  $\lambda_T$ . Consequently, clients require lower reputation to be classified as *trusted*, resulting in false negatives (i.e., malicious clients rated as *trusted*). At the same time, high initial reputation scores require that honest clients show high reputation to be classified as *trusted*. Setting the initial reputation scores too high may thus result in false positives, i.e., honest clients classified as either *indefinite* or even *untrusted*. In order to balance this tradeoff, we therefore choose  $R_0=R'_0=50$  in the remainder of this chapter.

Furthermore, TrustMeter shows better performance when the clients apply the random-fair exchange strategy as compared to the other exchange strategies. This difference is due to the number of triplets exchanged in each strategy. Clients applying the realistic strategy exchange more triplets than using the other strategies, since all triplets are exchanged. In the random-unfair strategy, only a subset of triplets is exchanged, while clients agree on the number of triplets to exchange in the random-fair strategy. As shown in Section 5.3.2, fewer triplets are exchanged when clients apply the random-fair strategy. By exchanging fewer packets, honest clients are less impacted by the reduction in reputation caused by not yet identified malicious clients they have met. Consequently, their reputation scores reflect better their actual behavior that allows a better assessment of their trust level.

#### Reputation Thresholds ( $\lambda_U$ and $\lambda_T$ )

The reputation thresholds determine the reputation ranges associated to each trust level as illustrated in Table 7. By design, these thresholds are centered on the initial reputation scores, i.e., 50 (cf. Section 6.4.2). Figure 22 presents the temporal evolution of the MCC for the different pairs of reputation thresholds.

For all exchange strategies, TrustMeter shows *poor* results for the threshold pair (0,100), as the MCC values remain equal to -1 for the entire time of the simulation. Using this threshold pair, clients can only be classified as *indefinite* that exclusively results in false positives and false negatives. For the pair (10,90), the performance of TrustMeter improves, as TrustMeter can also classify clients as *trusted* and *untrusted*. To reach these categories, clients however need to show either a very high reputation score (i.e., greater than 90) or a very low reputation score (i.e., lower than 10). As a result, only few clients reach such thresholds and thus, the remaining clients are classified as *indefinite*, leading to high percentages of false positives and negatives. By narrowing the threshold pair around values of 50, TrustMeter identifies malicious clients faster. The best results are reached for the pair (50,50) with which there are only two possible trust levels, *trusted* and *untrusted*. As there is no *indefinite* trust level in this case, new clients starting with an initial reputation of 50 may rapidly be classified as *untrusted* as soon as they meet a spammer not yet identified. This would definitely prevent them from contributing to the path jumbling. As a result, initial reputation scores greater than 50 should be chosen in order to support the future contribution of newly active clients, if the pair (50,50) is selected. Under these conditions, the detailed results show that TrustMeter correctly distinguishes honest from malicious clients in most cases, increasing the percentages of true positives and true negatives.

Overall, the random-fair exchange strategy shows better results for the same threshold pair than the other strategies. For example, at the end of the simulation, the MCC values



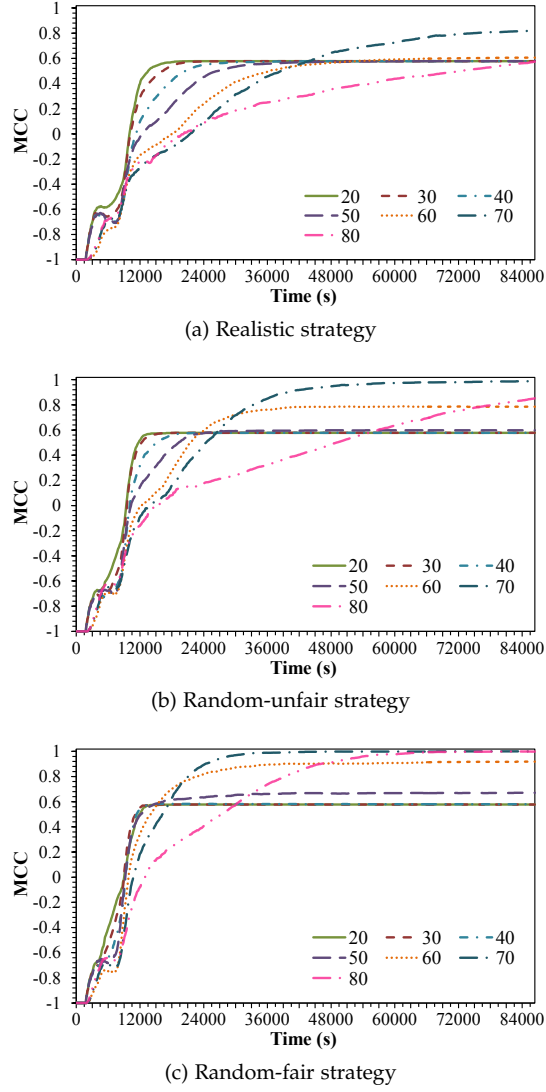


Figure 21: MCC for different values of the initial reputation scores  $R_0$  and  $R'_0$  (with  $R_0 = R'_0$ ) and exchange strategies over time

for the pairs (30,70), (40,60), and (45,55) are around 0.60 for the realistic strategy, 0.63 for the random-unfair strategy, and 0.68 for the random-fair strategy. In particular, TrustMeter reaches a MCC of 0.63 at then end of the simulation for the pair (10,90) with the random-fair strategy as compared to only 0.04 and -0.35 for the random-unfair and realistic strategies, respectively. As detailed in Section 6.4.2, this difference is due to the number of triplets exchanged in each strategy. The lower the number of triplets exchanged, the lower the reduction of reputation caused by encountered malicious clients who are not yet identified as such. Consequently, honest clients can reach  $\lambda_T$  faster when applying the random-fair strategy and be hence correctly classified.

In summary, the interval size from the center should not be too large, so that honest clients can still reach  $\lambda_T$  and malicious clients drop below  $\lambda_U$  in a realistic time. Simultaneously, the interval should be large enough to allow honest clients to prove their genuine behavior before being classified as *untrusted* and not be allowed to exchange triplets with others. Based on the above results, we therefore choose  $\lambda_U=30$  and  $\lambda_T=70$  as reputation thresholds in order to balance this tradeoff.

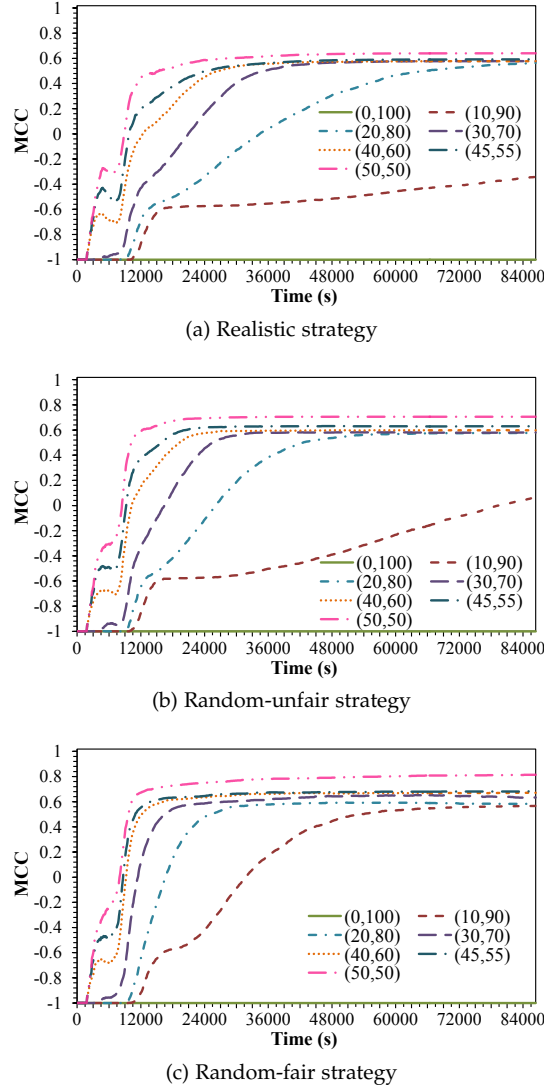


Figure 22: MCC for different pairs of reputation thresholds  $\lambda_U$  and  $\lambda_T$  and exchange strategies over time

#### Delivery Timeout ( $T_{\text{delivery}}$ )

As detailed in Section 6.2.2, clients set a timeout  $T_{\text{delivery}}$  that triggers the retransmission of triplets, which have not reached the server yet. In this section, we study the influence of the duration of this timeout on the performance of TrustMeter in identifying attackers. Additionally, we investigate the corresponding latency until triplets reach the application server. Figure 23 illustrates the temporal evolution of the MCC for different timeout durations, while Figure 24 illustrates the corresponding average latency. For the smallest timeout values, TrustMeter shows the worst performance for all exchange strategies, as shown, e.g., by a MCC value of -0.08 for the realistic strategy, 0.03 for the random-unfair strategy, and 0.30 for the random-fair strategy at the end of the simulation for a timeout of 1,500 s. With a short timeout duration, clients tend to prematurely consider the exchanged triplets as dropped, even if they may have been further exchanged or not yet reported to the server. This causes a reduction in reputation of the clients with which they exchanged these triplets. As a result, these may be incorrectly classified as *untrusted* despite their honest behavior. In order to reduce the percentage of false positives, the timeout should hence be chosen long enough to at least allow clients to report the triplets after the exchange before considering them as dropped. At the same time, selecting a long timeout delays the identification of actual droppers. Thus, it negatively impacts on the function of the collaborative path hiding mechanism.

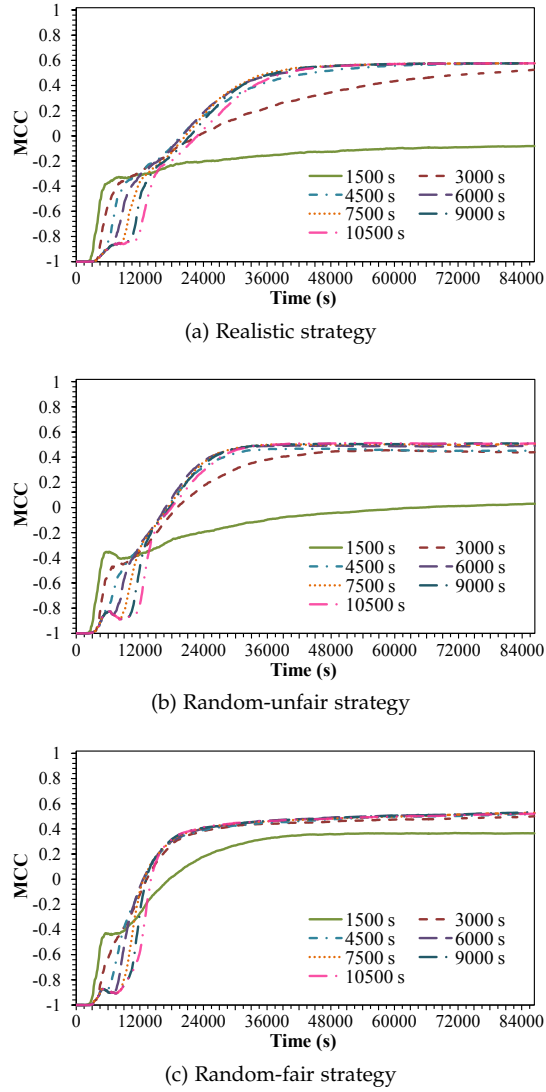
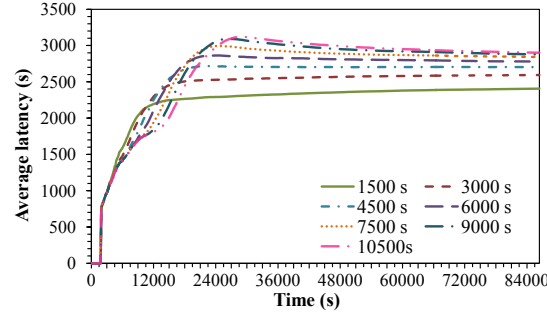
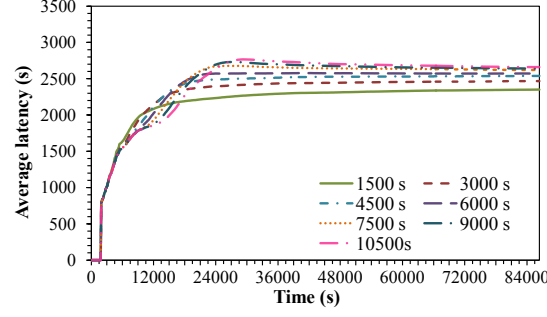


Figure 23: MCC for different durations of the delivery timeout  $T_{\text{delivery}}$  and exchange strategies over time

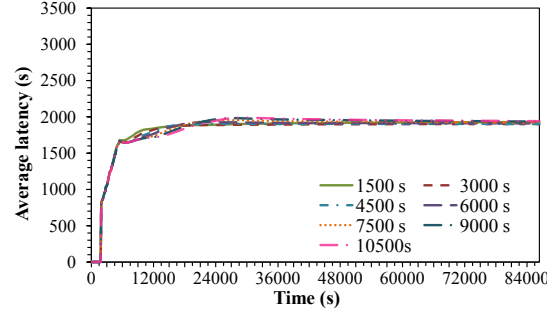
Furthermore, the latency increases with the timeout duration when the clients apply the realistic and random-unfair exchange strategies, as shown in Figure 24a and Figure 24b, respectively. Additionally, the averaged latency is shorter for the random-fair strategy as compared to the random-unfair and realistic strategies. For example, for a timeout duration of 1,500 s, the latency is equal to around 1,900 s for the random-fair strategy, 2,350 s for the random-unfair strategy, and 2,400 s for the realistic strategy. This difference is again due to the number of exchanged triplets for each strategy. Using the random-fair strategy, clients exchange fewer packets with others. This means that they report a higher fraction of their own collected triplets directly to the server. Consequently, the latency is mainly determined by the period of reporting, i.e., one hour on average in our scenario, and the duration of timeout has thus only a limited impact on the latency as confirmed in Figure 24c. In contrast, a larger number of packets are exchanged using the random-unfair and realistic strategies. This hence increases the probability that triplets are exchanged over several hops, thus resulting in an increased latency. In summary, the shorter the timeout, the shorter the latency for the realistic and random-unfair strategies. Simultaneously, a short timeout increases the amount of retransmitted triplets, and hence introduces additional overhead. Note that we quantify this overhead in Section 6.5. In order to balance both aforementioned tradeoffs, we therefore select a timeout of 7,500 s.



(a) Realistic strategy



(b) Random-unfair strategy



(c) Random-fair strategy

Figure 24: Average latency until packets reach the server for different durations of the delivery timeout  $T_{\text{delivery}}$  and exchange strategies over time

#### 6.4.3 Identification of Droppers

We first investigate the performance of TrustMeter in identifying droppers with different percentages of droppers, dropping rates, and attack scenarios. Note that the same analysis for spammers is conducted in Section 6.4.4. In the following, we assume the following values for the system parameters: (1)  $R_0=R'_0=50$ , (2)  $\lambda_U=30$  and  $\lambda_T=70$ , and (3) 7,500 s as timeout.

##### Percentage of Droppers

In this scenario, we assume that the droppers launch their attacks at the start of the simulation and continue to drop all triplets. Figure 25 represents the temporal evolution of the MCC for different percentages of droppers. It shows that the latency until TrustMeter succeeds in identifying most droppers depends on both the percentage of droppers as well as the exchange strategy applied by the users. Overall, the lower the percentage of droppers, the faster their identification, as the reputation of all clients from the triplets generators to the droppers are affected by the dropping action. The fewer droppers, the higher increase in reputation, allowing honest clients to reach the *trusted* level faster. The identification of droppers is faster

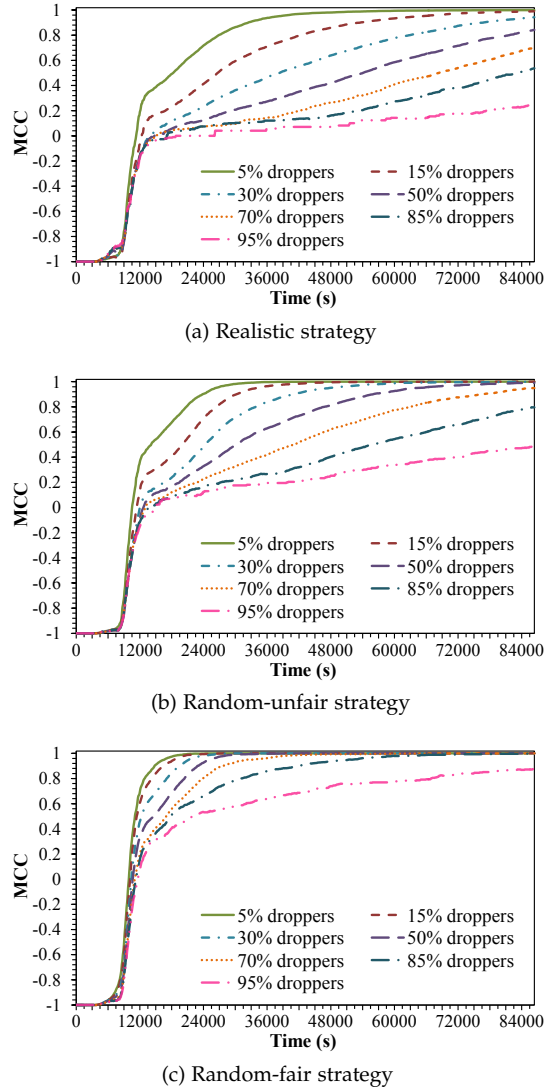
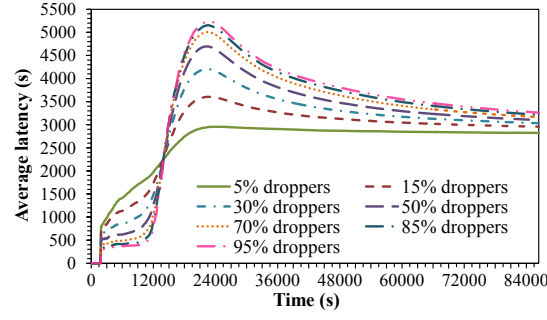


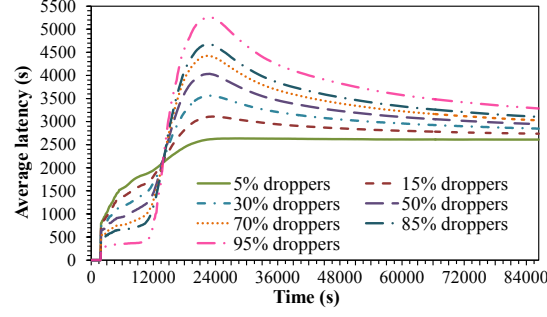
Figure 25: MCC for different percentages of droppers and exchange strategies over time

for the random-fair strategy with 21,000 s (i.e., around six hours and ten minutes) for 5% of droppers, as compared to the random-unfair and realistic strategies with 34,800 s (nine hours and 20 minutes) and 59,500 s (16 hours and 28 minutes), respectively. This means that only around six reports are necessary for the droppers' identification with the random-fair strategy, as reporting occurs every hour in average. Since clients exchange fewer triplets with the random-fair strategy, the impact of triplets dropped by others on the reputation of honest clients is lower than with the other strategies.

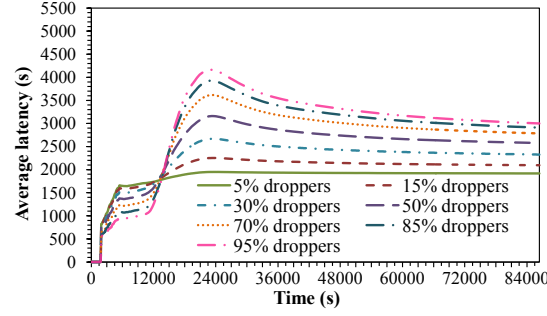
Furthermore, Figure 26 confirms that the latency increases with the percentage of droppers. For example, Figure 26b shows a difference in latency of up to 2,400 s between 5% and 95% of droppers at  $t=22,800$  s. However, the latency decreases again when the droppers have been identified by the server, as honest clients do not exchange triplets with them anymore. For 95% of droppers, the latency decreases from around 1,980 s using the realistic strategy (see Figure 26a), to 1,620 s and 1,140 s using the random-unfair (see Figure 26b) and random-fair strategy (see Figure 26c), respectively. The impact of droppers on the latency is lower for the random-fair strategy, since the droppers are identified and excluded faster than with the other strategies as shown in Figure 25. Simultaneously, the decrease in latency takes longer with the random-fair exchange strategy, as the clients exchange fewer triplets. If the triplets are not



(a) Realistic strategy



(b) Random-unfair strategy



(c) Random-fair strategy

Figure 26: Average latency until triplets reach the server for different percentages of droppers and exchange strategies over time

directly reported by clients, the probability that they are exchanged is lower and the path to the server takes longer.

### Dropping Rate

Next, we assume a population of 90 honest clients and ten droppers and investigate the impact of the dropping rate on the performance of our scheme. Figure 27 shows that TrustMeter performs *poorly* in distinguishing honest clients from droppers for dropping rates below and equal to 30%. For these dropping rates, droppers drop fewer packets that results in a slower reduction in reputation, and hence a longer time span until they are considered as *untrusted*. However, the performance improves for 50% dropping rate with MCC values around 0.60 for the realistic strategy (see Figure 27a), 0.40 for the random-unfair strategy (see Figure 27b) and 0.20 for the random-fair strategy at  $t=86,400$  s (see Figure 27c). For dropping rates over 70%, TrustMeter reaches *perfect* identification at the end of the simulation. As previously, the identification of droppers is globally faster when the clients apply the random-fair strategy as compared to the other exchange strategies.

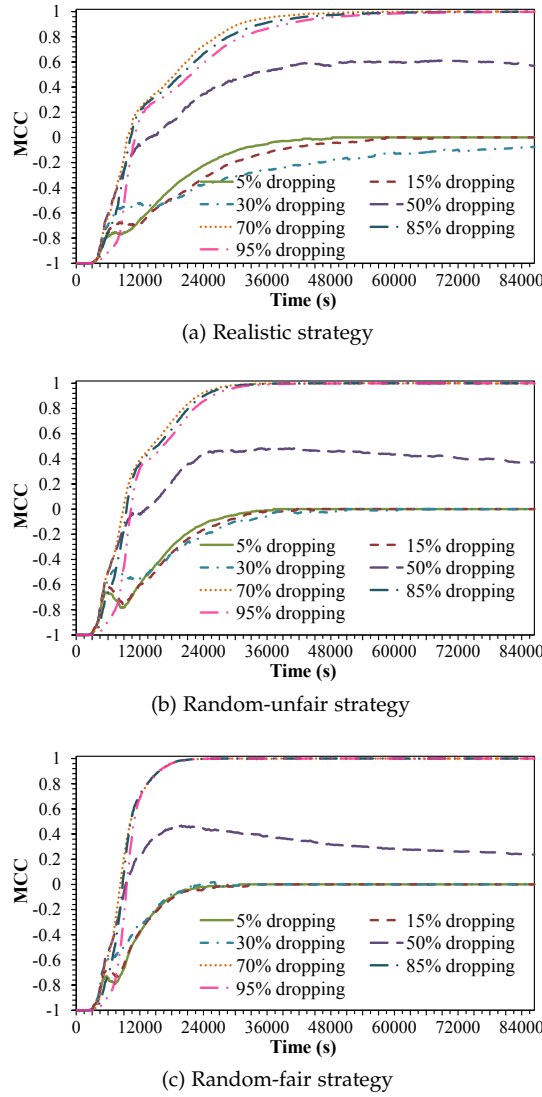


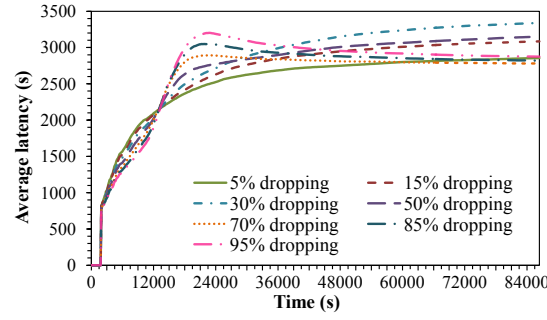
Figure 27: MCC for different dropping rates and exchange strategies over time

Figure 28 shows the dependency of the latency on the correct identification of the droppers. Again, the latency decreases when droppers are identified (i.e., for dropping rates higher than 70%). This is particularly visible for the realistic strategy, where the latency diminishes from 3,200 s to 2,800 s for 95% dropping (see Figure 28a). Moreover, droppers have less impact on the latency when the clients apply the random-fair and random-unfair strategies, since clients exchange fewer triplets with them and report more triplets to the server directly. This also explains the overall difference in latency between the different exchange strategies.

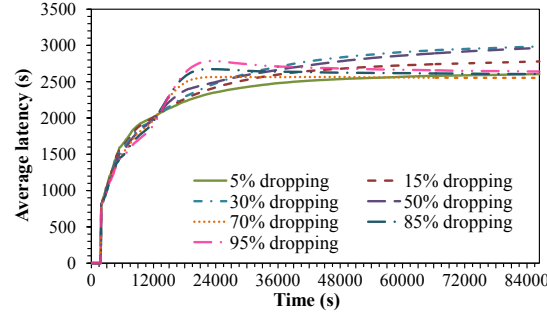
#### Distribution of Dropping Attacks

In the following, we assume a constant population of 30 droppers that drop all packets according to different temporal patterns and 70 honest clients. By considering the following scenarios, we aim at analyzing the impact of the time at which the droppers start their attacks on the performance of TrustMeter in identifying droppers. This provides insights about the influence of potential reputation gained by the droppers before the attack on their identification as *untrusted* clients. Additionally, we examine the impact of the number of droppers simultaneously launching dropping attacks.

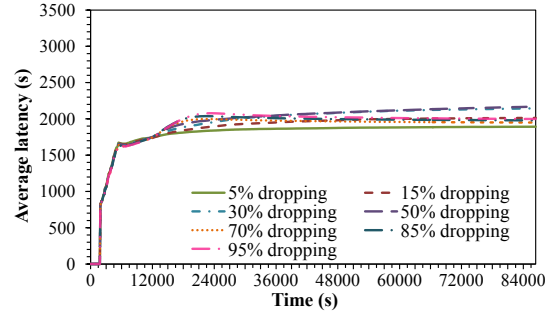
- ▷ SCENARIO 1: All 30 droppers start at  $t=0$  s (reference scenario).



(a) Realistic strategy



(b) Random-unfair strategy



(c) Random-fair strategy

Figure 28: Average latency until triplets reach the server for different dropping rates and exchange strategies over time

- ▷ SCENARIO 2: All 30 droppers start at  $t=10,800$  s.
- ▷ SCENARIO 3: All 30 droppers start at  $t=21,600$  s.
- ▷ SCENARIO 4: 15 droppers start at  $t=10,800$  s and the 15 others start at  $t=21,600$  s (i.e., two waves of 15 droppers each).
- ▷ SCENARIO 5: Ten droppers start at  $t=10,800$  s, ten others at  $t=21,600$  s, and the remaining at  $t=32,400$  s (i.e., three waves of ten droppers each).
- ▷ SCENARIO 6: Six droppers start at  $t=10,800$  s, six others at  $t=16,200$  s, six others at  $t=21,600$  s, six others at  $t=27,000$  s, and the last six at  $t=32,400$  s (i.e., five waves of six droppers each).

Figure 29 represents the temporal evolution of the MCC for the different dropping scenarios and exchange strategies. The results show that the later the attack is launched, the longer TrustMeter needs to correctly identify the droppers. For the random-fair strategy, all droppers are correctly identified 24,600 s after the attack was launched in scenario 1, while detection took 46,000 s in the scenario 2, and over 64,800 s in the scenario 3 (see Figure 29c). In other words,



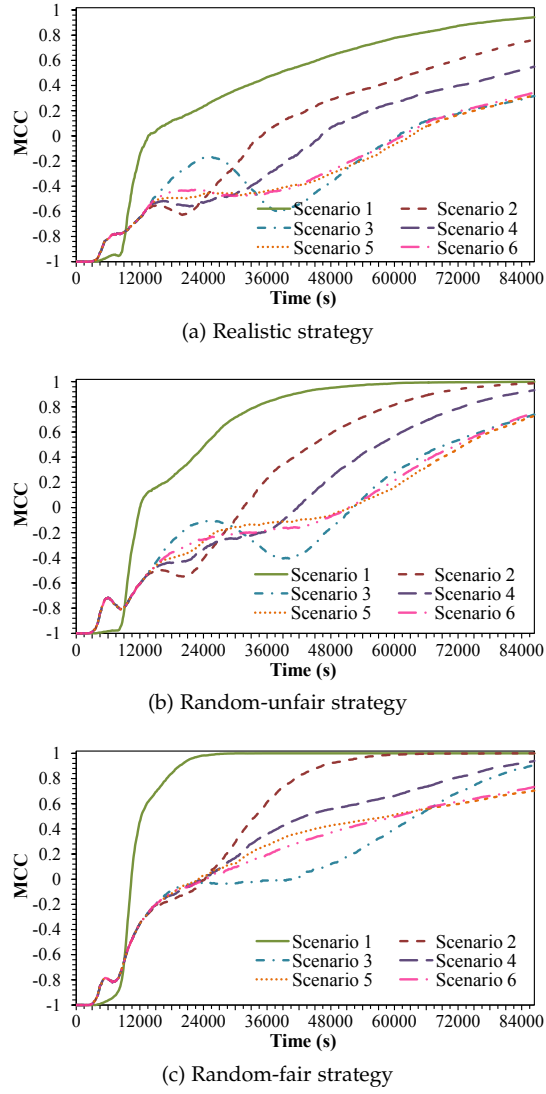


Figure 29: MCC for different dropping attack scenarios and exchange strategies over time

the longer droppers act as honest users, the longer it takes to TrustMeter to unmask them after their attack. Since droppers are able to first build reputation, this is the time necessary for their reputation to drop below  $\lambda_U$ . The more reputation gained, the longer the delay. Furthermore, the identification is faster when clients apply the random-fair and random-unfair strategies. By applying these exchange strategies, clients statistically exchange fewer triplets than with the realistic strategy. This means that their reputation is less impacted by exchanges with malicious clients as discussed previously. Simultaneously, the increase in reputation depends on the number of exchanged triplets. The fewer triplets exchanged, the lower the reputation increases. Consequently, droppers are able to build less reputation when applying these exchange strategies and thus, lower reduction in reputation is necessary to drop below  $\lambda_U$ . At the same time, the number of exchanged triplets may not be sufficient to significantly reduce the reputation of droppers that have launched their attacks, potentially increasing the identification delay. However, the results show that the reduction in reputation is sufficient to mitigate the small number of exchanged triplets and caters for the identification of the droppers in a reasonable time. Concerning scenarios 4 to 6, TrustMeter performs better for scenario 4, since the attacks are launched earlier as compared to the other scenarios discussed above. For scenarios 5 and 6, there are only few differences between the performance of TrustMeter in identifying droppers even if an additional attack is launched in scenario 6. In

both scenarios, TrustMeter is still in the initial phase in which it progressively starts to identify spammers, leading to only few differences in its performance.

#### 6.4.4 Spammers

We analyze the performance of TrustMeter in identifying spammers under the same attack scenarios as above.

##### *Percentage of Spammers*

We assume that the spammers launch their attacks at the start of the simulation and continue to spam all triplets. Figure 30 illustrates the temporal evolution of the MCC for different percentages of spammers. For all exchange strategies, the performance of TrustMeter improves with the number of spammers until reaching the best performance for 50% of spammers at the end of the simulation. The lower the number of spammers, the higher the probability that they exchange spam with honest clients. As a result, the reputation of honest clients decreases, leading to their incorrect classification as *indefinite*. Moreover, the reduction in reputation increases with the number of exchanged spam triplets. Hence, TrustMeter performs better when the clients apply the random-fair strategy for up to 50% of spammers as shown by, e.g., a MCC value of 0.63 at the end of the simulation for 5% spammers (cf. Figure 30c), as compared to 0.26 for the random-unfair strategy (cf. Figure 30b) and 0.00 for the realistic strategy (cf. Figure 30a). The probability that spammers exchange spam with other spammers increases with the number of spammers. Their reputation hence diminishes, leading to their correct identification. However, the performance of TrustMeter decreases from 50% of spammers. Since the amount of injected spam increases, the probability that honest clients relay this spam increases. Simultaneously, the impact of the exchange strategy diminishes when the number of spammers increases, as the probability to exchange spam increases.

##### *Spamming Rate*

Next, we assume a population of ten spammers and consider different spamming rates. Figure 31a shows that TrustMeter only *fairly* performs in identifying spammers when clients apply the realistic strategy, as clients exchange more triplets than with the other strategies. Therefore, the reputation of honest clients getting spam decreases more rapidly, leading to their incorrect categorization. However, when honest users are identified as *untrusted*, they are temporarily quarantined and can only upload previously exchanged or their own triplets. If the uploaded triplets are correct, clients rapidly regain reputation that causes the change of their trust level and allow them to exchange triplets again. Otherwise, the clients are maintained in quarantine. Depending on the duration of the quarantine, the privacy of the clients may be endangered if clients only report their own collected triplets. In this case, the clients can apply an additional privacy-protection mechanism, such as replacing the exact location included in the triplet by a coarser one. In comparison, the performance of TrustMeter improves for the random-unfair strategy due to a lower amount of exchanged triplets, and hence a lower impact on their reputation if the triplets are spam. TrustMeter shows the best results for the random-fair strategy with up to a MCC value of 0.85 at the end of the simulation for 100% spamming (cf. Figure 31c). Moreover, the improvement of the performance of TrustMeter with the spamming rate for both the random-unfair and random-fair strategies is due to the values of  $\lambda_U$  and  $\lambda_T$  that determine the reputation range for the different categories in which clients are classified. With our settings, the reduction in reputation caused by a low spamming rate may not be sufficient to reach  $\lambda_U$ . The spammer may hence be incorrectly classified as *indefinite* or *trusted*. The higher the spamming rate, the lower the increase in reputation, and hence the higher probability that the spammers are correctly classified.

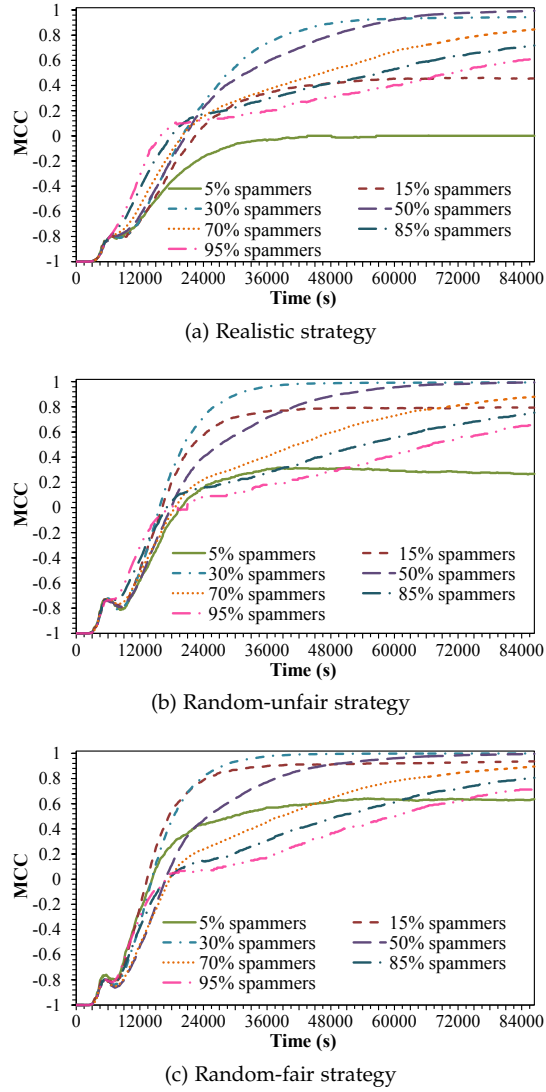


Figure 30: MCC for different percentages of spammers and exchange strategies over time

#### Distribution of Spamming Attacks

We finally assume a constant population of 30 spammers launching their attacks according to the same scenarios as presented in Section 6.4.3. Scenarios 1 to 3 correspond to the same number of spammers launching their attacks at  $t=0$  s,  $t=10,800$  s, and  $t=21,600$  s, respectively. Similarly to droppers, Figure 32 shows that the longer malicious clients can build reputation, the longer it takes for TrustMeter to unmask them after they launched their attacks. However, TrustMeter requires more time to correctly identify all spammers as compared to droppers, since all spammers are still not identified at the end of the simulation for the random-fair strategy (cf. Figure 32c), whereas it is reached at  $t=57,600$  s for the droppers (see Figure 29c). Additionally, the impact of the applied exchange strategy on the performance of TrustMeter is the same as for droppers. Indeed, TrustMeter performs better when clients apply the random-fair strategy as compared to the random-unfair and realistic strategies. Since fewer packets are exchanged in the random-fair strategy, the reputation built by the spammers before launching their attack is lower. Consequently, a lower reduction in reputation is necessary to drop below  $\lambda_U$  and thus, be identified as *untrusted*.

Furthermore, scenarios 4 to 6 correspond to smaller groups of spammers that simultaneously launch their attack at different times. Scenario 4 consists of two groups of 15 spammers, while scenarios 5 and 6 consist of three groups of ten spammers and five groups of six spammers,

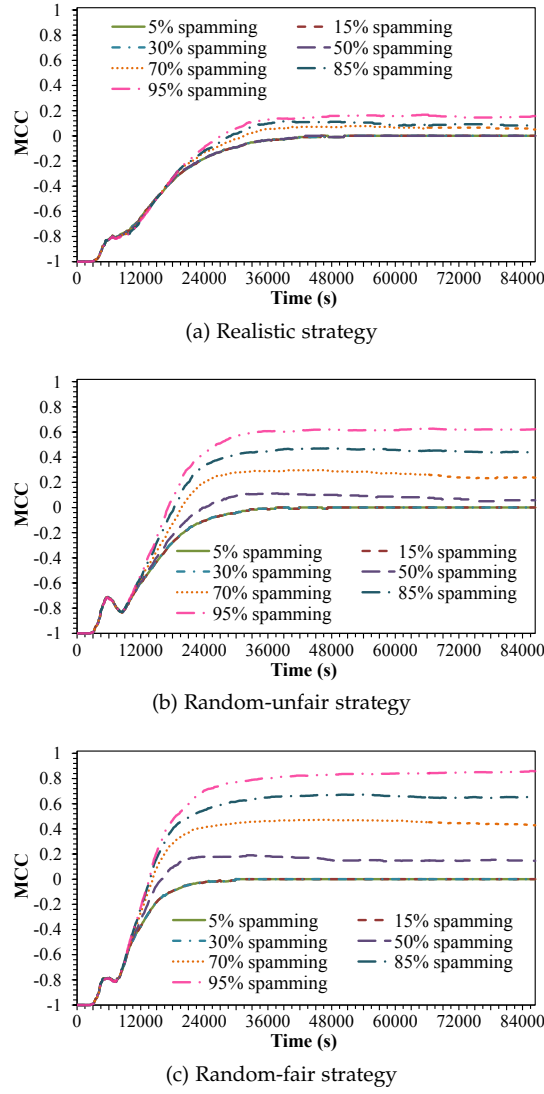


Figure 31: MCC for different spamming rates and exchange strategies over time

respectively. Figure 32 illustrates that TrustMeter identifies spammers faster in scenario 4, as the attacks are launched earlier when compared to the other scenarios. Moreover, there are almost no differences between the results for scenarios 5 and 6. In these scenarios, additional attacks (with smaller number of spammers) are launched after those in scenario 4. However, we have previously observed that the later the attacks, the longer TrustMeter requires to correctly identify both droppers and spammers. Overall, TrustMeter also requires more time to identify spammers as compared to droppers. This means that the presented results still correspond to the initial phase in which TrustMeter progressively identifies the spammers.

#### 6.4.5 Evaluation Summary

The performance of TrustMeter in identifying malicious users can be considered *good* to *excellent* in most cases. In some of the cases, *perfect* identification has even been reached around the end of the simulation. Due to the multi-hop nature of the scheme, its performance improves with time and TrustMeter succeeds in identifying a majority of malicious clients after only few reports sent by the clients to the server on a hourly basis.

Overall, TrustMeter performs better in identifying malicious clients when users apply the random-fair strategy. By applying this strategy, the users statistically exchange fewer triplets

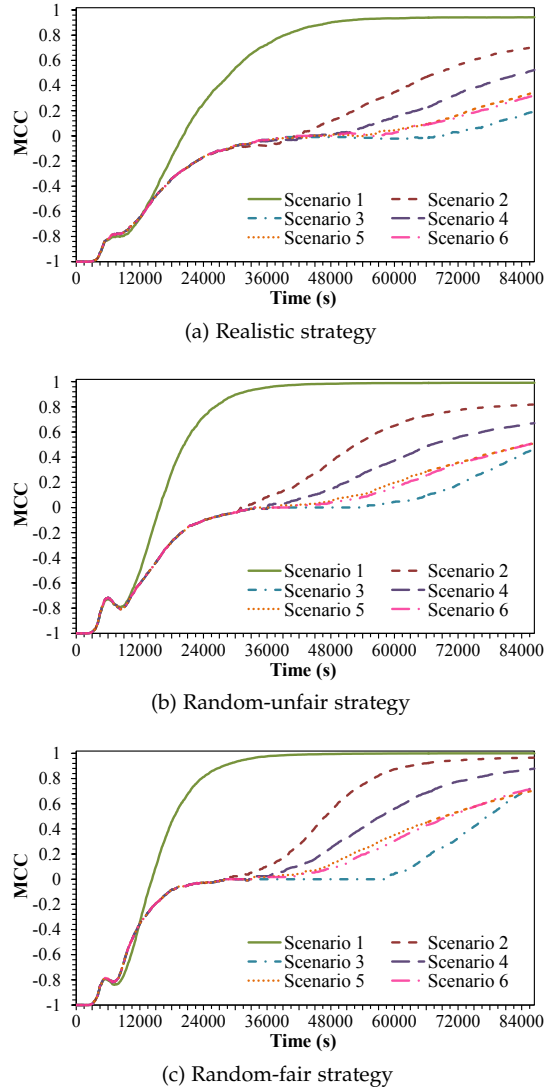


Figure 32: MCC for different spamming attacks scenarios and exchange strategies over time

with other users. In case of exchanges with malicious users not yet identified as such, the consequences for honest users are less severe, since fewer triplets are dropped and fewer spam triplets are exchanged. Especially the identification of spammers when users apply the realistic strategy is difficult, since honest users can exchange all their genuine triplets with a not yet recognized spammer in the worst case. If these users directly upload the exchanged triplets to the server, they will be consequently identified as spammers. Honest users have thus less chance of being falsely identified when applying the random-fair strategy compared to the random-unfair and realistic strategies.

We have however seen in the preceding chapter that applying the random-fair strategy leads to a better identification of the jumbled triplets by curious application administrators. A tradeoff therefore exists between the performance in identifying the exchanged triplets as well as malicious users. The more difficult the identification of exchanged triplets for the application server, and hence the better privacy protection, the worse the detection of malicious users.

There is an additional tradeoff between the responsiveness of TrustMeter in identifying malicious users and the frequency of exchanges between users, feedback transmitted to the application server as well as reported triplets. The less frequent the exchanges and reports, the longer it takes to TrustMeter to distinguish malicious users from honest ones. Simultaneously,

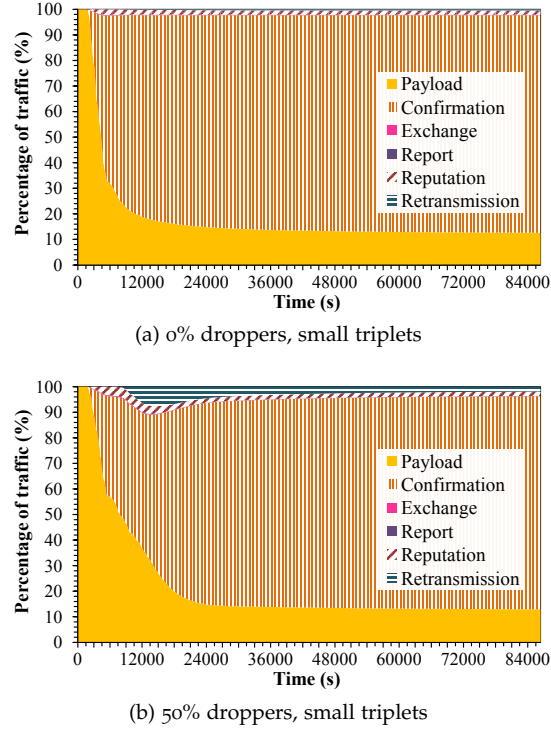


Figure 33: Temporal evolution of the overheads for different number of droppers and small triplet size

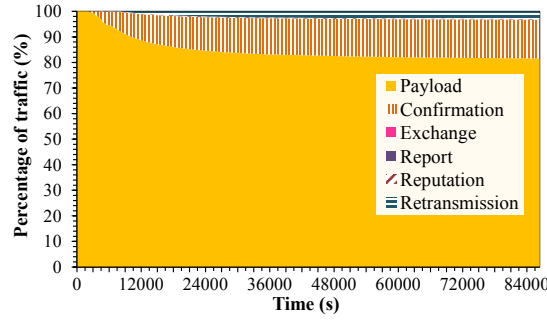
reporting triplets more frequently, i.e., after fewer meetings, allows the application server to potentially identify more exchanged triplets, and hence reduce the privacy protection again as shown in the preceding chapter.

## 6.5 TRAFFIC OVERHEAD

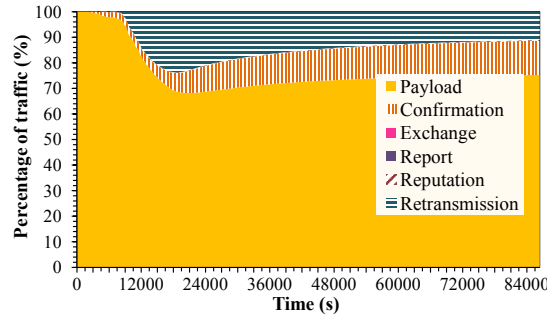
Next, we quantify the traffic overheads incurred by both TrustMeter and potential attacks. To this end, we assume that TrustMeter is integrated into two distinct participatory sensing applications. These applications differ in the size of the collected triplets in order to compare the introduced overheads to the traffic generated by the application. In particular, the first application collects primitive sensor readings (e.g., accelerometer data or sound levels) with sizes between 100 bytes to 1 kilobytes, whereas the second application collects rich sensor readings (e.g., short sound samples) with sizes around 16 kilobytes [CRKH11]. We further assume that 100 clients apply the realistic exchange strategy and that the percentage of active droppers that drop all triplets is 0% and 50%, respectively.

Figure 33 and Figure 34 represent the temporal distribution of the following traffic classes over 24 hours for small triplet size and large triplet size, respectively. *Payload* corresponds to triplets exchanged between clients. *Exchange* and *Report* consist of the lists of client IDs and the corresponding number of triplets exchanged and reported, respectively. *Retransmission* corresponds to triplets that are resent by the client after the delivery timeout expired. *Confirmation* are the lists of triplet IDs reported to the server, while *Reputation* consists of the reputation values for each client computed by the server.

After an initial bootstrapping phase, 80% of the total traffic is caused by the *confirmation* class for triplets of small sizes as shown in Figure 33. In comparison, the *payload* and *reputation* classes only represent 15% and 2% of the traffic, respectively. This difference is due to the length of the triplet IDs that are currently 32 bytes long using a SHA-256 hash function. Moreover, for each triplet reported, the server transmits its ID to all clients. The overhead generated by the triplet retransmissions increases with the number of droppers, but remains under 4% of the total traffic. Similarly to the latency (see Section 6.4.3), the overhead



(a) 0% droppers, large triplets



(b) 50% droppers, large triplets

Figure 34: Temporal evolution of the overheads for different number of droppers and large triplet size

diminishes, once droppers have been identified after 13,600 s. The overheads for the remaining classes are negligible. For triplets of large sizes, the overhead caused by the confirmations represents only 15% of the traffic, whereas the exchange of triplets generates 80% of the traffic as shown in Figure 34a. Since the triplets are larger, their retransmission also generates more overhead. This overhead reaches over 15% of the traffic in the case of 50% droppers. Note that the distributions of the traffic classes in presence of spammers are the same as illustrated in Figure 33a and 34a, as their attacks do not trigger retransmissions.

In summary, confirmation lists generate a large overhead for applications gathering primitive sensor readings. In order to reduce this overhead, a trivial solution could be to reduce the size of the triplet IDs. However, this would imply using a hash function producing shorter digests. This increases the risk that the server can infer the identifiers of the clients having generated the triplets. As a result, a tradeoff should be found between a sufficient length for the triplet IDs to ensure the confidentiality of the client identifier and short enough to keep the overhead small.

## 6.6 RELATED WORK

Assessing the trust of clients exchanging and reporting sensor readings have not been addressed in such settings in the context of participatory sensing applications. For example, reputation systems have been presented in [HKH10a], [YZR11], and [CRH<sup>+</sup>12a]. However, they mainly concentrate on evaluating the correctness and quality of the reported sensor readings. While we integrate such mechanisms to identify spammers, additional mechanisms are necessary to assess the contribution of clients in the collaborative path hiding mechanism. Therefore, we specially focus on solutions tailored to delay tolerant and opportunistic networks [LKBGo6], since they share the more similarities with our system model detailed in Section 6.1.1.

Existing solutions for opportunistic networks can be classified into two categories, depending on the nature of the information used to compute the levels of trust. For example, [BCPo8], [TLA10], and [CBCC10] leverage social relationships, while [MSCo9] and [KTH10] rely on the



numbers of encounters between users. In particular, users explicitly indicate their degree of trust for other users based on their social relationships and build a web of trust in [TLA10]. However, leveraging social relationships limits the number of exchange partners to a subset of users sharing strong social links. This hence restrains the efficacy of the collaborative path hiding scheme. Another model proposed in [CBCC10] combines information about social relationships and quality of service metrics. However, this model is designed to ensure the delivery of packets between clients using the shortest path. In contrast, in our approach, the destination of the triplets is the application server and clients are encouraged to exchange triplets with other users before reporting them to the server in order to increase the jumbling of the triplets and improve the mutual privacy protection of the clients.

Models presented in [MSC09, KTH10] solely determine the trust relationships between users based on the number of their encounters. In [MSC09], users generate a shared secret at each encounter. The shared secret is then used at the next encounter to prove that a previous encounter between the same users already happened. As a result, the trust relationship between users strengthens at each exchange of shared secret. In comparison to our approach, this scheme does not take into account the behavior of the users, but focuses on the frequency of their meetings. Similarly, the model proposed in [KTH10] is not based on the generation and exchange of shared secrets, but analyzes the location of the users over long period in order to determine similar behaviors. Users exhibiting similar behavior are assumed to share stronger trust relationships than those showing different behavior. Relying only on encounters to assess the degree of trust in other users is insufficient in our scenario, since malicious clients can visit densely populated locations in order to rapidly build strong trust relationships, before launching dropping attacks.

In summary, existing mechanisms in opportunistic networks are not designed to assess the contribution of clients in the exchange and reporting of triplets as defined in our system model. Novel mechanisms, such as presented in TrustMeter, therefore need to be introduced.

## 6.7 SUMMARY

In this chapter, we have presented TrustMeter, an application-agnostic scheme that assesses the behavior of the users contributing to the collaborative path hiding process. It computes and attributes a trust level to each user based on statistics provided by the users about past exchanges of sensor readings annotated with spatiotemporal information. Malicious users can be clearly identified based on their low trust ratings and subsequently excluded from the data jumbling process. At the same time, the ratings provide as little information as possible about the exchanges in order to preserve the provided privacy protection. We have implemented the TrustMeter scheme and investigated its performance in identifying malicious users in a realistic application scenario. By means of extensive simulations, we have analyzed various attack scenarios and shown that the performance of TrustMeter is good to excellent in most scenarios. Overall, TrustMeter shows better performance in identifying droppers as compared to spammers after only few reports sent by the clients to the server. We have further discussed the robustness of TrustMeter against threats to reputation and quantified the overheads introduced by the proposed scheme under diverse scenarios. In particular, we have shown its practicability in participatory sensing applications that gather rich sensor readings with around 30% additional traffic incurred. Using our approach, users hence obtain the assessed trust level of encountered users and can subsequently determine the exchange strategy to apply based both on the provided trust levels and their personal preferences in terms of, e.g., privacy protection as studied in Chapter 5.



## INCOGNISENSE: AN ANONYMITY-PRESERVING REPUTATION FRAMEWORK

---

It takes many good deeds to build a good reputation, and only one bad one to lose it.

B. Franklin

As illustrated in Chapter 3 and especially in Chapter 5, existing privacy-preserving solutions often rely on breaking the associations between consecutive user contributions. However, these associations are required by reputation systems applied in participatory sensing applications to weed out incorrect user contributions as detailed in Chapter 4. In this chapter, we present our framework called IncogniSense [CRH<sup>+</sup>12a, CRH<sup>+</sup>13] that addresses the aforementioned inherent conflict between privacy and reputation. Our system utilizes periodic pseudonyms generated using blind signatures and relies on reputation transfer between these pseudonyms. In particular, our approach does not rely on a trusted third party to ensure the privacy protection of the users. Users can parametrize our system according to their individual privacy preferences.

We first introduce our assumptions and threat model in Section 7.1, before presenting the IncogniSense framework in Section 7.2. We analyze the robustness of IncogniSense against threats to reputation in Section 7.3 and conduct a multi-dimensional evaluation of IncogniSense in Section 7.4. We further study the feasibility of our approach in Section 7.5. We finally compared IncogniSense to existing related work in Section 7.6, before making concluding remarks in Section 7.7.

### 7.1 ASSUMPTIONS AND THREAT MODEL

In this section, we present our threat model and detail our assumptions. We consider that our adversary set includes malicious clients, application servers, and the Reputation and Pseudonym Manager (RPM) (described in Section 7.2). Again, the adversaries follow the Dolev-Yao threat model [DY83], i.e., they are able to listen to all communication, fabricate, replay, and destroy messages. They are, however, not able to break cryptographic mechanisms.

#### 7.1.1 Threats to Reputation

The goal of the adversaries, primarily malicious clients, is to corrupt the reputation system in order to artificially increase their own reputation. Self-promotion can be achieved by means of Sybil attacks, in which the malicious client creates an arbitrary number of identities that vouch for each other [Dou02]. Alternatively, malicious clients may replay old messages to gain reputation without contributing new sensor readings.

We assume that the application server and the RPM are protected against fraudulent access by well-established security mechanisms. Hence, adversaries are not able to access stored data, or change the behavior of applications. Denial of service attacks are considered beyond the scope of this work.

#### 7.1.2 Threats to Anonymity

Another goal of the adversaries is to breach the anonymity of the clients. Adversaries track the interactions of the clients with the reputation system and attempt to establish relationships between successive pseudonyms and link them to a unique real identity. We assume that the

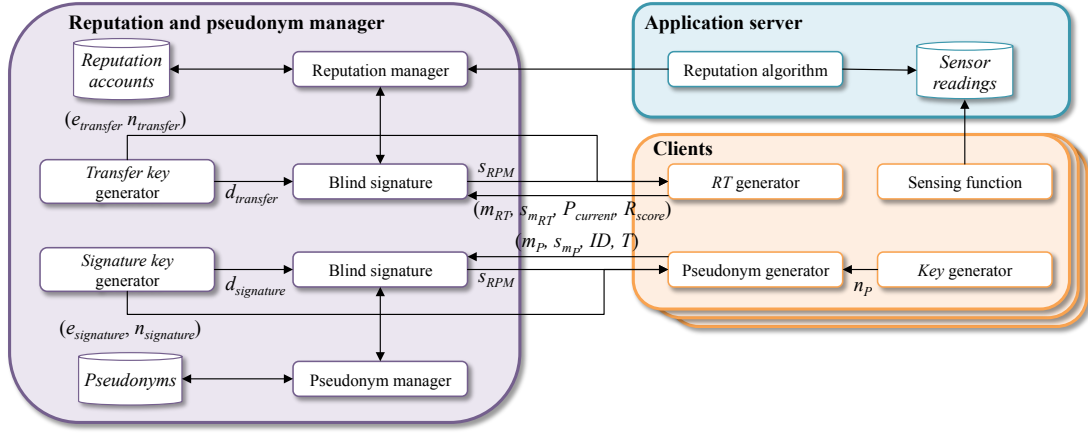


Figure 35: Overview of the IncogniSense framework

reported sensor readings do not contain any direct indication of the identity of the clients and that the interactions of the clients with the application server are anonymized using, e.g., disposable IP and MAC addresses and anonymous communication networks [SCP<sup>+</sup>10].

## 7.2 THE INCOGNISENSE FRAMEWORK

We first provide an overview of the IncogniSense framework and detail the underlying mechanisms. Note that further details on the utilized blind signatures are available in Appendix A.1 to A.3. We then present our reputation cloaking mechanisms and highlight the differences between IncogniSense and the framework proposed in [MR07], which forms the basis for parts of our work.

### 7.2.1 Overview and Underlying Mechanisms

The IncogniSense framework illustrated in Figure 35 includes clients, an application server, and a RPM (introduced in Section 7.1). We assume a participatory sensing application in which several active clients collect sensor readings and report them to the application server. In this scenario, our framework relies on two main mechanisms: (1) the utilization of periodic pseudonyms by the clients to report the collected sensor readings, and (2) the transfer of reputation scores between consecutive pseudonyms in order to conserve the reputation gained by the clients across multiple time periods. In particular, the proposed solution is comprised of the following four steps, which are repeated sequentially for each time period  $T$ .

#### Pseudonym Generation

We assume that each client has a permanent identifier  $ID$ , a private key  $PR$ , a public key  $PU$ , and is registered with the RPM. Clients generate pseudonyms in association with the RPM and based on blind signatures [Cha83]. By employing blind signatures, the objective is two-fold: (1) to ensure that each client has a unique pseudonym for each  $T$  in order to prevent Sybil attacks (cf. Section 7.1.1), and (2) to hide the generated pseudonym from the RPM in order to prevent it from linking the pseudonym to the client's real identity. In fact, blind signatures ensure the authenticity of signed messages without revealing their content to the signing entity and prevent the signing entity from linking the message content with the identity of its creator. For each pseudonym, each client generates a new key pair  $(PR_P, PU_P)$ , with  $PR_P$  the private key and  $PU_P$  the public key. Note that the public key is available to all, while the private key is only known by the client who generated it. Next, the client hands over the public key to the RPM for blind signature as detailed in Appendix A.2. For each time interval  $T$ , the RPM uses a different private key  $PR_{signature}$  in the blind signature that determines the period of validity

of the pseudonym. Moreover, the RPM only signs one pseudonym per client per time period in order to prevent Sybil attacks. As a result, the client uses the blindly signed pseudonym and the newly generated private key to report sensor readings to the application server and thus, to transfer reputation to its subsequent pseudonym. If the RPM signs more than one new pseudonym in each  $T$ , the application server and the RPM cannot link the reported sensor readings to the real identity of the client.

#### *Reporting of Sensor Readings*

Within  $T$ , the client periodically reports sensor readings to the application server using its current pseudonym  $P_{\text{current}}$  (i.e., the pseudonym valid in  $T$ ). The application server verifies the validity of the pseudonym with the RPM, evaluates the sensor reading using a reputation model, and attributes a reputation score  $R_{\text{score}}$  to  $P_{\text{current}}$ , with  $R_{\text{score}} \in \mathbb{Z}$ . The application server eventually makes use of the reputation scores to identify inaccurate contributions. Depending on the selected reputation model, the application server can discard the contributions with low reputation scores or reduce the weighting for such contributions in the computation of summaries. Note that the design of both reputation algorithm and model is considered beyond the scope of this thesis. Existing solutions such as [HKH10a] can, however, be easily integrated into our generic framework. Next, the application server transmits  $R_{\text{score}}$  to the RPM, which maintains a reputation account for each pseudonym and adds  $R_{\text{score}}$  to  $P_{\text{current}}$ 's reputation account.

#### *Generation of Reputation Tokens*

Before the expiration of  $P_{\text{current}}$  (i.e., at the end of  $T$ ), the client generates its next pseudonym  $P_{\text{next}}$  for the next time period as described in Section 7.2.1. It then transfers the gained reputation with  $P_{\text{current}}$  to  $P_{\text{next}}$  in order to conserve it after the expiration of  $P_{\text{current}}$ . The reputation transfer is realized using Reputation Tokens (RTs) generated by the clients in collaboration with the RPM to prevent reputation corruption. The transfer process also makes use of blind signatures with  $P_{\text{next}}$  being the blinded content in order to prevent the RPM from linking the client's consecutive pseudonyms. In fact, each client withdraws reputation scores from  $P_{\text{current}}$ 's account and then, deposits them using RTs in  $P_{\text{next}}$ 's account, both accounts being maintained by the RPM. For each generated RT, the client using  $P_{\text{current}}$  indicates the reputation score to be transferred in the RT to the RPM as detailed in Appendix A.3. The RPM verifies the balance of the reputation account of  $P_{\text{current}}$  and decrements it accordingly. The RPM has different key pairs referred to as *transfer keys*, each of them being associated to a different reputation value. It thus uses the key pair corresponding to the reputation score to be transferred on the RT to blindly sign it. Consequently, the signing key (and not the content of the RT) determines the reputation value of each RT that prevent clients from manipulating RTs' value. However, the linking between  $P_{\text{current}}$  and  $P_{\text{next}}$  is only prevented if more than one RT is generated per  $T$ . Otherwise, the linking is straightforward as there is only a single reputation transfer between two pseudonyms.

#### *Reputation Transfer*

The client registers with the RPM using  $P_{\text{next}}$  within the current  $T$  and hands over RT(s). The RPM verifies that each RT has not been used before and credits the reputation account of  $P_{\text{next}}$  from RT's value.

#### 7.2.2 Cloaking Mechanisms

In theory, the utilization of blind signatures prevents the linking of consecutive pseudonyms. However, in practice, the reputation transfer between two consecutive pseudonyms may reveal insights about their succession. For example, consider the case where  $P_{\text{current}}$  has the highest reputation among all pseudonyms. Now assume that after reputation transfer, this same

reputation score is associated with the pseudonym,  $P_{next}$ . It is fairly straightforward for an adversary to establish a link between  $P_{current}$  and  $P_{next}$ . An adversary able to track the reputation scores over several time intervals can link pseudonyms used in different periods. To prevent such attacks, we propose that the clients cloak (i.e., alter) their reputation scores before their transfer. Since the RPM prevents unjustified reputation promotion by controlling the generation of the RTs, the clients only transfer reputation scores lower or equal to their actual reputation. While cloaking the reputation scores prevents a reputation analysis attack, it may cause degradation in reputation score since it entails adding a perturbation to the same. In this section, we present three different reputation cloaking schemes, which address this tradeoff in different ways.

#### *Floor Function Reputation Transfer (Floor)*

This scheme divides the entire spectrum of reputation scores into fixed intervals and classifies the clients into these intervals based on their reputation scores. For the actual transfer, the clients use the floor value of the reputation interval as reputation score and transfer it using a single RT. Note that we use the floor value to prevent unjustified reputation promotion. Similar to the concepts of  $k$ -anonymity [Swe02], the scheme forms groups of pseudonyms sharing the same reputation score. The larger the groups, the more indistinguishable the pseudonyms, and the harder it is for the adversary to link consecutive pseudonyms. The anonymity protection is thus determined by the size of the groups, which depends on the selection of the intervals. Selecting large intervals may increase the number of pseudonyms within a group, but may negatively affect the reputation scores due to the coarse granularity of the chosen intervals. The size of the reputation intervals is therefore an important design parameter for addressing the tradeoff between anonymity and reputation.

#### *Transfer of Random Sets from Reputation Partition Sets (RandSet)*

*RandSet* divides the reputation score to be transferred into multiple RTs based on a predefined set partitioning (e.g., {10, 50, 250}) used by all users. For each reputation transfer, the division of the reputation score into the set partitioning is determined by each client individually. For example, a client can partition a reputation score of 70 into one RT of value 50 and two RTs of value 10, or alternatively use seven RTs of value 10. The client then randomly selects one or more RTs handed over to the RPM. We refer to  $p$  as the probability that an individual RT is used to transfer reputation. In other words, each RT is discarded and not transferred with a probability  $1 - p$ . Both the division and discarding mechanisms increase the entropy of the actually transferred reputation and prevent the linkage between consecutive pseudonyms. However, discarding RTs can reduce the reputation scores. The reduction in reputation is linearly correlated with the size of the RTs, i.e., discarding a large RT leads to greater loss in reputation. In summary, both set partitioning and discarding probability influence the tradeoff between anonymity and reputation.

#### *Transfer of Random Scores from Reputation Partition Sets (RandScore)*

The initial reputation score is split into different RTs of predefined size as in *RandSet*. During the transfer, all the RTs are transferred. However, the transferred score of the individual RTs is lowered according to a random function. The set partitioning impacts the entropy of the transferred scores and the loss in reputation. The range of the possible scores linearly increases with the size of the RTs. Simultaneously, the transferred reputation statistically diminishes, as the probability of transferring the initial RT values decreases, while the size of the RTs increases.

Note that the spectrum of possible cloaking mechanisms is not limited to the aforementioned schemes. We have specially selected these schemes based on the diversity of their key parameters to address the tradeoff between anonymity protection and reputation degradation.

### 7.2.3 Comparison with the “Reputation using Pseudonyms” Framework

We contrast IncogniSense with the Reputation using Pseudonyms (RuP) framework proposed in [MR07], since our work improves the presented ideas. In particular, we highlight shortcomings of the RuP algorithm in terms of cryptographic overhead and vulnerability to reputation manipulation and argue how IncogniSense addresses these issues differently.

#### Algorithmic Differences

In the RuP algorithm, the client includes the interval of validity of the pseudonym in the blind signature. Each client creates  $n$  pseudonyms to generate a sole valid pseudonym. The  $n$  pseudonyms are transmitted to the RPM, which randomly selects  $n - 1$  pseudonyms and requests the client to send the corresponding random values  $\frac{1}{r}$ . The RPM then encrypts the  $n - 1$  pseudonyms and verifies that they are valid for the same interval. If the verification is successful, it signs the  $n$ th pseudonym, which becomes the valid pseudonym. The generation of RTs also necessitates that the client prepares  $n$  messages and the RPM verifies  $n - 1$  messages before signing the  $n$ th message. In IncogniSense, we decouple the interval of validity of the pseudonym and the value of the reputation to transfer from the blind signatures by introducing periodic *signature keys* and different *transfer keys* for each value of RT, reciprocally.

#### Cryptographic Overhead

Table 8 summarizes the cryptographic overhead for generating pseudonyms associated with both schemes. In [MR07], the client generates  $n$  key pairs, selects  $n$  values  $r$  and  $\frac{1}{r}$ , and executes  $n$  encryptions to generate one valid pseudonym. The generated key pairs of non-selected pseudonyms should not be reused to prevent the RPM from linking the pseudonyms to the client’s identity. In IncogniSense, the client prepares one blind signature per pseudonym, i.e., it generates only one key pair and encrypts one message, and the RPM verifies only one signature per interval and per client (instead of  $n - 1$ ), which significantly reduces the computational overhead if we assume a large base of clients. Similar conclusions can be drawn for the RT generation, except that no key pair is generated by the client.

#### Reputation Manipulation

In the RuP algorithm, the RPM cannot verify the interval of validity of the pseudonym included in the blinded message. It randomly verifies  $n - 1$  of the  $n$  generated pseudonyms and signs the  $n$ th, hoping that it contains the same validity interval as the verified pseudonyms. This implies that malicious clients can generate multiple pseudonyms for a given time interval with a probability of  $\frac{1}{n}$ . If such an attack is successful, then the adversaries can seriously compromise the reputation system (see Section 7.1.1). Likewise, malicious clients can generate fraudulent RTs and increase their reputation. IncogniSense completely eliminates the possibility of Sybil attacks and also prevents adversaries from compromising the reputation transfer process. The utilization of periodic *signature keys* and the verification of already existing pseudonyms by the RPM guarantees that each client has a unique pseudonym per interval. Similarly, the utilization of different *transfer keys* allows the RPM to easily verify and guarantee the value of the transferred reputation, preventing malicious clients from generating falsified RTs.

In summary, IncogniSense achieves better protection against reputation manipulation, while significantly reducing the cryptographic overhead for the client.

## 7.3 ROBUSTNESS AGAINST THREATS TO REPUTATION

Next, we revisit the assumptions and threat model presented in Section 7.1 and show that IncogniSense is resilient against the following attacks.

Table 8: Comparison of the cryptographic overhead of the RuP framework and IncogniSense

	CLIENT		RPM	
	RuP [MR07]	IncogniSense	RuP [MR07]	IncogniSense
Generated key pairs	n per interval	1 per interval	1	1 per interval
Prepared messages	n per interval	1 per interval	-	-
Signature verifications	-	-	n-1 per interval and client	0
Number of signatures	-	-	1 per interval and client	1 per interval and client

### 7.3.1 Sybil Attacks

Malicious clients can attempt to generate multiple pseudonyms for a given interval to increase their reputation through cross-recommendations. However, IncogniSense is protected against these attacks since the RPM maintains a list of the clients that have presented pseudonyms for blind signature along with their validity interval. Once a pseudonym has been signed for a given interval, the RPM discards all other pseudonyms submitted by the same client.

### 7.3.2 Replay Attacks

Malicious clients may attempt to replay old messages for the following two reasons: (1) artificially increase their reputation by replaying RTs, (2) debit the reputation account of honest clients without the victims receiving the associated credit by replaying the message to be signed. The RPM, however, maintains a list of IDs associated with each RT. The RPM can thus detect malicious clients trying to corrupt the reputation system and prevent both attacks. Malicious clients are prevented from forging RTs and manipulating the reputation of other clients, as the creation of the RTs requires the signature of the corresponding pseudonym, which is only possible using the pseudonym's private key.

### 7.3.3 Manipulation of Reputation Accounts

Malicious clients can attempt to alter the reputation stored in the RPM. The clients, however, do not have direct access to their reputation accounts and the RPM is protected against unauthorized access using standard cryptographic primitives. As such, this attack cannot be mounted.

### 7.3.4 Reporting of Falsified Sensor Readings

Malicious clients can try to report falsified sensor readings on behalf of others to degrade their reputation. IncogniSense protects honest clients against this attack by requesting clients to authenticate with both the application server and the RPM. These entities verify the validity of the pseudonyms before considering their contributions and/or delivering them information. Such an attack would only be successful if malicious clients can access the private keys of the targeted clients and those of their respective pseudonyms, which is beyond the scope of our attacker model.

In summary, we have shown that IncogniSense is robust by design against all threats directed against the reputation system by malicious clients.

## 7.4 ANONYMITY PROTECTION

We analyze the resilience of IncogniSense against the threats to anonymity identified in Section 7.1.2. In particular, we measure the level of anonymity, i.e., unlinkability, that can be achieved by the different reputation cloaking schemes introduced in Section 7.2.2. In a



Table 9: Probability distribution of transient reputation scores

Score attribution probability	0.25	0.35	0.25	0.10	0.05
Attributed reputation scores	10	5	0	-5	-10

first step, we assume that the reputation scores are lost if they are not transferred to the next pseudonym at the end of the period. We hence quantify the reduction in reputation score caused by the different cloaking approaches and investigate the trade-off between anonymity protection and loss in reputation. Next, we assume that the clients can store reputation for future use and analyze the impact of stored reputation scores on anonymity protection for both a constant and variable population of clients.

#### 7.4.1 Transient Reputation Scores

In the following, we assume that the reputation scores are transient, i.e., their validity duration is limited to the duration of the period in which they have been gained, and we adopt the following simulation setup and method.

##### *Simulation Setup and Method*

We implemented a simulator in Java to model the behavior of clients, RPM and application server. Each simulation run is for 100 time intervals  $T$ . We repeat each simulation 100 times and present the averaged results. All clients remain active for the duration of the simulation. During each time interval, the clients generate five random sensor readings and report them to the application server. The application server runs a simulated reputation algorithm, which randomly attributes reputation scores to the pseudonyms according to the distribution presented in Table 9. Note that the actual values of both sensor readings and reputation scores do not impact the performances of the cloaking schemes in terms of linkability and can thus be selected randomly. Moreover, we consider the study of additional reputation models and distributions as future work.

We adopt the assumptions of our threat model (see Section 7.1) and assume that the RPM and the application server are malicious internal observers. We further assume that adversaries collude and aim at linking consecutive pseudonyms. For each simulated interval, the RPM and the application server have access to the following information. The RPM observes the generated RTs, the utilized RTs, and the updates of the reputation accounts, while the application server observes the current pseudonyms, the initial and final reputation scores, and the updates of reputation scores.

Based on the observed information, the adversaries identify the set of pseudonyms active in each time interval. We refer to  $S_{P_{\text{current}}}$  as the set of pseudonyms active in a given time interval and  $S_{P_{\text{next}}}$  as the set of pseudonyms active in the subsequent time interval. As all clients remain active for the duration of the simulation,  $|S_{P_{\text{current}}}| = |S_{P_{\text{next}}}|$  and there exists a bijection between the sets  $S_{P_{\text{current}}}$  and  $S_{P_{\text{next}}}$ . For each  $P_{\text{current}}$ , the adversaries first identify all potential successors and create links between  $P_{\text{current}}$  and the identified  $P_{\text{next}}$ s based on their reputation scores and the created and used RTs as detailed in Algorithm 5. Note that the second condition in step 3 ( $S_{RT_{\text{used}}} \subseteq S_{RT_{\text{created}}}$ ) is verified if the values of the  $RT_{\text{used}}$ s are lower than those of the  $RT_{\text{created}}$ s if the *RandScore* scheme is applied. For the other schemes, the same condition is verified if the values are equal. Next, they iteratively eliminate the created links based on the bijection between  $S_{P_{\text{current}}}$  and  $S_{P_{\text{next}}}$  according to Algorithm 6. Algorithm 6 first searches for single links between  $S_{P_{\text{current}}}$  and  $S_{P_{\text{next}}}$ . Due to the bijection between both sets, all other existing links of  $P_{\text{current}}$  and  $P_{\text{next}}$  are removed (step 6).  $P_{\text{current}}$  and  $P_{\text{next}}$  are then flagged (step 7) and the link between both pseudonyms is confirmed (step 8). This indicates that  $P_{\text{next}}$  has been identified as the successor of  $P_{\text{current}}$ . Flagged pseudonyms and confirmed links are then excluded from further searches. When

**Algorithm 5** Identification of potential successors for transient reputation scores

---

**Input:**  $R_{P_{current}}$ :  $P_{current}$ 's reputation,  $R_{P_{next}}$ :  $P_{next}$ 's reputation,  $S_{RT_{created}}$ : set of RTs created by  $P_{current}$ ,  $S_{RT_{used}}$ : set of RTs used by  $P_{next}$

```

1: for all  $P_{current} \in S_{P_{current}}$  do
2:   for all  $P_{next} \in S_{P_{next}}$  do
3:     if  $R_{P_{next}} \leq R_{P_{current}}$  and  $S_{RT_{used}} \subseteq S_{RT_{created}}$  then
4:       create a link between  $P_{current}$  and  $P_{next}$ 
5:     end if
6:   end for
7: end for

```

---

**Algorithm 6** Elimination of potential successors for transient reputation scores

---

**Input:**  $S_{Links}$ : set of links created in Algorithm 5

```

1: linkConfirmed  $\leftarrow$  true
2: while  $\exists$  non-flagged pseudonym  $\in (S_{P_{current}} \cup S_{P_{next}})$  and linkConfirmed = true do
3:   linkConfirmed  $\leftarrow$  false
4:   for all non-flagged pseudonym  $\in (S_{P_{current}} \cup S_{P_{next}})$  do
5:     if  $\exists$  a single link between  $P_{current}$  and  $P_{next}$  then
6:       remove all other links of  $P_{current}$  and  $P_{next}$ 
7:       flag  $P_{current}$  and  $P_{next}$ 
8:       linkConfirmed  $\leftarrow$  true
9:     end if
10:   end for
11:   if linkConfirmed = false then
12:     search for other non-flagged  $P_{current}$ s having the same potential successors
13:     if number of found  $P_{current}$ s = number of common  $P_{next}$ s then
14:       remove all other links of the found  $P_{current}$ s and  $P_{next}$ s
15:       flag the found  $P_{current}$ s and  $P_{next}$ s
16:       linkConfirmed  $\leftarrow$  true
17:       break search
18:     end if
19:   end if
20: end while

```

---

all existing single links have been successively removed, Algorithm 6 analyzes the remaining pseudonyms and searches for  $P_{current}$ s having the same potential successors (step 12). This search returns one or several subsets of  $S_{P_{current}}$  sharing the same successors. The algorithm then selects the smallest subset whose size is equal to the number of common successors (step 13). It removes all other links of these pseudonyms (step 14), flags the pseudonyms (step 15), and confirms the links (step 16). The analysis of sets of pseudonyms having the same successors is interrupted and the algorithm restarts at step 2 with the search for single links, which potentially appear while removing the previous links.

Finally, as a measure of the level of anonymity provided by the reputation transfer, we calculate how many potential successors  $x_k$  (i.e., links) a pseudonym  $k$  has on average. We express the amount of potential successors as the fraction of the total number of clients  $N = |S_{P_{current}}|$ . For each interval, we calculate this metric as shown in Equation 12.

$$\text{Metric} = \frac{1}{N} \cdot \sum_{k=1}^N x_k \quad (12)$$

A value of 1 implies perfect anonymity achieved during the transfer, since all pseudonyms within  $S_{P_{next}}$  are potential successors to each individual pseudonym of  $S_{P_{current}}$ . The lower the value, the smaller the set of pseudonyms that are the potential successors, which in turn implies that it may be easier to establish a link between consecutive pseudonyms. We can



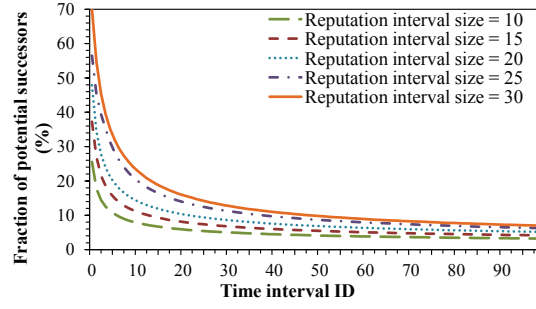
therefore directly assess the quality of the anonymization achieved by the cloaking schemes using this metric.

### Evaluation Results

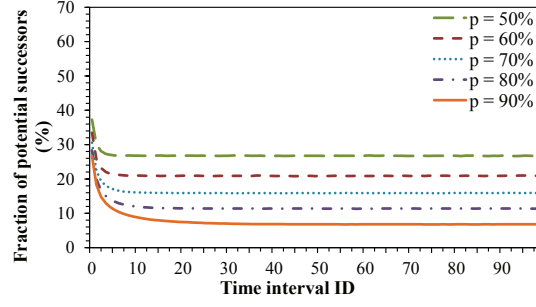
By applying the above setup and method, we first individually evaluate the reputation cloaking schemes (see Section 7.2.2). In particular, we analyze the effect of the most important configuration parameter relevant to each scheme and the resulting impact on the anonymity of the clients. Secondly, we compare how these schemes fare against each other. In this comparison, we use the transfer scheme used in [MR07] as a baseline. This scheme, also referred to as the *Full Reputation Transfer (Full)* scheme, always transfers the entire reputation in a single RT. It hence represents the worst case in terms of anonymity protection, since the reputation score remains the same for consecutive pseudonyms, which allows for easy linkability by the adversaries. Note that we compare the cloaking schemes based on selected variants that show a good tradeoff between anonymity protection and loss in reputation.

In the *Floor* scheme, the clients use the floor value of given reputation intervals as reputation score and transfer it using a single RT. Figure 36a illustrates the influence of different interval sizes (10, 15, 20, 25, 30) on the quality of the anonymity protection. The results show that the choice of large reputation intervals results in a higher level of protection, as more clients are grouped within the same interval and therefore become indistinguishable during the RT transfer. This protection comes at the expense of greater perturbation in reputation for some of the clients, since the difference between the actual reputation value and the floor of the interval also increases on average. Over time, the protection level decreases, since the reputation values of the individual clients spread over wider intervals as shown in Figure 38b for an interval size of 20. Figure 38 shows the development of reputations scores over time and is annotated with the 0.25- and 0.75-quantile of reputation scores of the clients for time interval 50. As a result, individual clients are more easily distinguishable in our attacker model, since potential successors can be identified more easily. The *Full* scheme shows a similar behavior (cf. Figure 38a), but is able to preserve higher reputation scores. This is illustrated in Figure 37, which shows the observed diminution in reputation scores per interval caused by the cloaking for selected variants of the proposed cloaking schemes that show a good tradeoff between anonymity protection and loss in reputation. With both schemes, clients are therefore easy to identify after a number of rounds since only a small number of peers have similar reputation scores.

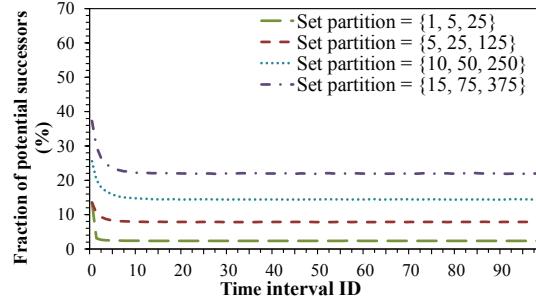
In the *RandSet* scheme, the reputation score is partitioned into different RTs of predefined size and a reputation transfer is performed with probability  $p$  for each RT. The partition in RTs is implemented as follows. The clients first generate as many RTs with the highest value of the set of partition as possible. Next, they use the second highest and so on, until the remaining reputation becomes lower than the lowest value of the set partition. Figure 36b illustrates the influence of the choice of  $p$  (50%, 60%, 70%, 80%, 90%) using the set partitioning {10, 50, 250} on the anonymity protection level. The fraction of potential successors stabilizes early, which means that attackers do not gain further evidence even if monitoring for extensive periods of time. This can be attributed to the reputation scores, which are not constantly increasing over time, but fluctuate in a certain interval as shown in Figure 38c for a probability  $p$  of 80%. Indeed, at a certain time instant the average increase in reputation due to the contribution to the application equals the average loss due to the discarding of reputation during the transfer. In contrast, the *Full* and *Floor* schemes constantly increase the reputation values as shown in Figures 38a and 38b. The percentage of potential successors using *RandSet* lies between 7% and 26% for  $p$  equal to 90% and 50%, respectively (see Figure 36b). The degree of anonymity protection increases with decreasing  $p$ , i.e., increasing discarding probability  $1 - p$ . Figure 37 shows the preserved reputation score for  $p$  equal to 80%. Obviously, for increasing  $p$ , the preserved reputation per round increases likewise. Note that the preserved reputation stays slightly below  $p$ , since the partitioning of the sets might not exactly fit the exact reputation value.



(a) Floor: Fraction of potential successors over time for selected sizes of reputation interval



(b) RandSet: Fraction of potential successors over time for different discarding probabilities  $1 - p$  and a partition set of {10, 50, 250}



(c) RandScore: Fraction of potential successors over time for different set sizes and a reduction in reputation between 0% and 50%

Figure 36: Level of unlinkability for reputation transfer schemes Floor, RandSet, and RandScore for a constant population of 100 clients and transient reputation scores

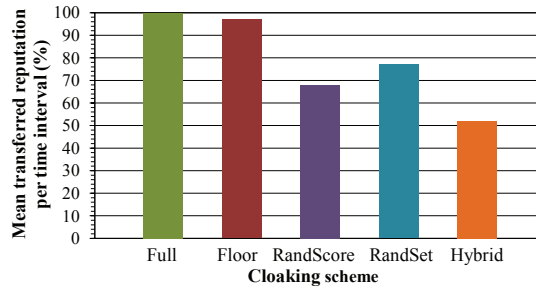
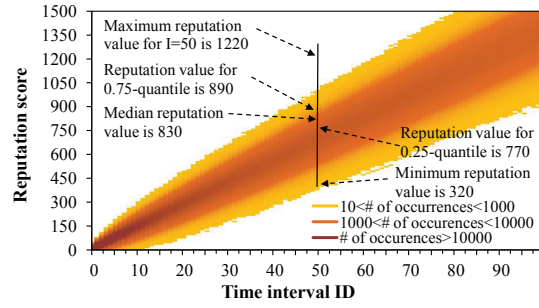
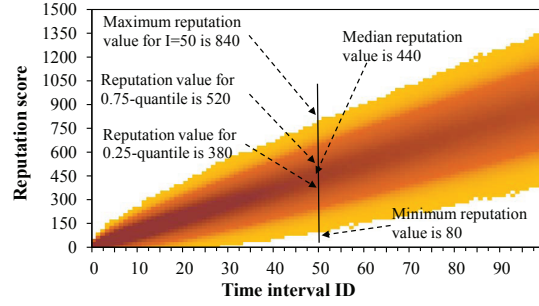


Figure 37: Comparison of transferred reputation for a constant population of 100 clients and transient reputation scores. *Floor* uses an interval size of 20. *RandScore* uses a partition set of {10, 50, 250} and  $p=75\%$ . *RandSet* uses  $p=80\%$ . *Hybrid* uses a partition set of {10, 20, 250} and  $p=80\%$

In the *RandScore* scheme, the reputation score is also partitioned into several RTs, which are all used in the reputation transfer. The transferred score is, however, lowered according



(a) Full reputation transfer



(b) Floor function reputation transfer with an interval size of 20

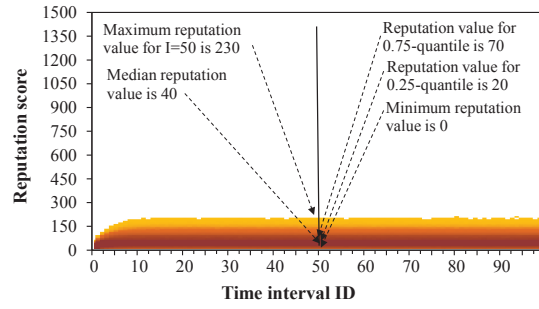
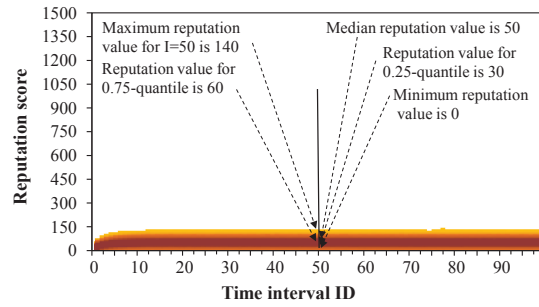
(c) Random set reputation transfer with  $p=80\%$ (d) Random score from set reputation transfer with a partition set of {10, 50, 250} and  $p=75\%$ 

Figure 38: Distribution of reputation score amongst clients over time for a constant population of 100 clients and transient reputation scores

to a random function. We analyze the influence of different set partitions while discarding between 0% and 50% (following a continuous uniform distribution) of the reputation score of a RT. Figure 36c shows that the level of anonymity protection increases with the set partitioning moving towards larger RT values. The smaller the lowest RT size in a set {1, 5, 10, 15}, the worse the protection of anonymity, since individual clients can now be tracked by the number of RTs exchanged (but not the random reputation score exchanged) as detailed

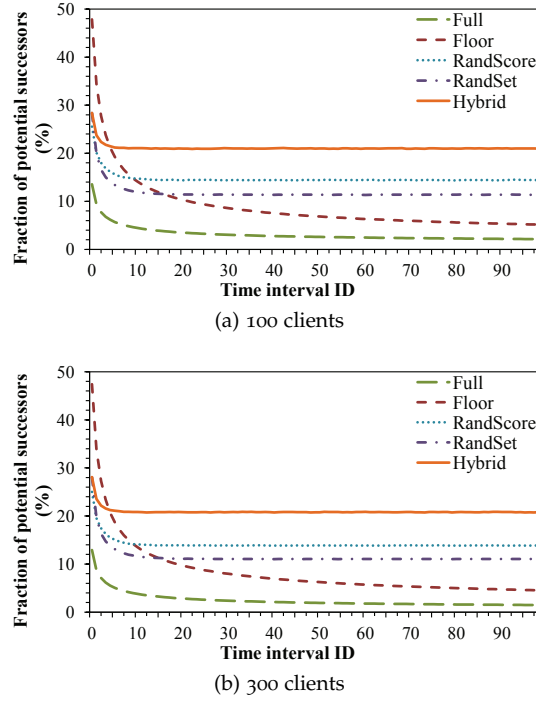


Figure 39: Scheme comparison and impact of the number of clients on the percentage of successors for transient reputation scores. *Floor* uses an interval size of 20. *RandScore* uses a partition set of {10, 50, 250} and  $p=75\%$ . *RandSet* uses  $p=80\%$ . *Hybrid* uses a partition set of {10, 20, 250} and  $p=80\%$ .

in Algorithm 6. Similar to *RandSet*, the performance quickly stabilizes since the reputation values inherently stay within certain bounds as shown in Figure 38d (for a partition set of {10, 50, 250} and a discarding probability of 25%), which makes identification very hard even for sophisticated attackers. Note that *RandSet* and *RandScore* perform similarly, if the same partitioning is applied and  $p$  equals the mean of the score discarding probability used in *RandScore*. Additionally, *RandSet* behaves similarly to the *Full* scheme for very low discarding probabilities ( $< 1\%$ ). In other words, reputation continuously increases for very low discarding probabilities. In contrast, reputation stabilizes within reputation bounds for higher probabilities. The stabilization time, however, depends on the discarding probability: The higher the probability, the faster the stabilization. For example, the reputation reaches its upper bound after 80 time intervals for a discarding probability of 5%.

For comparison purposes, we select one variant of each scheme that demonstrated a good tradeoff between anonymity protection and reputation loss based on the previous analysis. Hence, the following results are only representative for these variants. The parameterization is as follows.

- ▷ *Floor* utilizes an interval size of 20.
- ▷ *RandScore* partitions the RT sets into {10, 50, 250} and has a mean value of the discarding probability of 25%.
- ▷ *RandSet* retains RTs with a probability  $p$  of 80% and utilizes a similar partitioning {10, 50, 250}.

To model a realistic participatory sensing scenario, we investigate populations of 100 and 300 clients. We present the results in Figure 39 for all proposed reputation cloaking schemes for both scenarios under consideration. We include an additional scheme referred to as *Hybrid*, which combines the *RandSet* and *RandScore* schemes. The *Hybrid* scheme utilizes the partitioning {10, 50, 250} and retains RTs with a probability  $p$  of 80%. The reputation

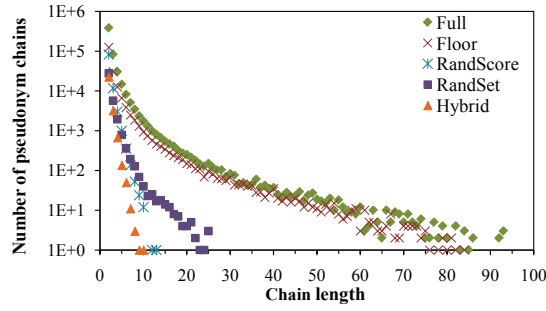


Figure 40: Number of identified pseudonym chains for a constant population of 100 clients and transient reputation scores. *Floor* uses an interval size of 20. *RandScore* uses a partition set of {10, 50, 250} and  $p=75\%$ . *RandSet* uses  $p=80\%$ . *Hybrid* uses a partition set of {10, 20, 250} and  $p=80\%$

score of the retained RTs is then randomly lowered as in the *RandScore* scheme. The *Hybrid* scheme is robust against observers that can track the number of RTs or the reputation scores obtained/transferred, since it cloaks both values during transfer. As seen from both graphs in Figure 39, all schemes are robust when increasing the client base from 100 to 300. In summary, the *Hybrid* scheme provides the best anonymity protection followed by *RandScore*, *RandSet*, *Floor*, and the *Full* scheme. *Hybrid*, *RandScore*, and *RandSet* provide a constant protection faster than *Full* and *Floor* due to the constantly increasing reputation observed for the latter schemes.

Figure 40 shows the number and the length of the pseudonym chains, i.e., the number of identified consecutive pseudonyms, for all schemes. Note that chains of length two represent identified links between two consecutive pseudonyms. Schemes that allow for long pseudonym chains are thus weaker than schemes that only allow the attacker to identify short chains. The *Hybrid* scheme is the most resilient scheme against attacks since it only allows adversaries to build very short chains of unique predecessor-successor relations (always shorter than 10). The *RandScore* (no chains longer than 13) and the *RandSet* (no chains longer than 25) have a vast majority of unlinkable clients, which is in line with our previous observations of a good anonymity protection. The *Floor* and the *Full* schemes perform worst in terms of identified chain length and number of identified chains. In a few cases, they allow adversaries to identify chains of up to 84 and 93 successive pseudonyms, respectively. Since the reputation scores with *Full* and *Floor* constantly increase, it is much easier for an attacker to follow individual pseudonyms. As a countermeasure, one could limit the maximum reputation score a client can obtain. The introduction of an upper bound would help to keep well behaving clients together in terms of reputation scores, thus making it much harder for an attacker to build pseudonym chains.

The anonymity of the clients is, however, protected at the price of a reduction in reputation as shown in Figure 37. Clients utilizing the *Hybrid* scheme suffer the highest loss in reputation, followed by the *RandScore* and the *RandSet* schemes. The *Full* and the *Floor* schemes manage to transfer nearly the complete reputation scores.

In summary, we have demonstrated that some of the presented schemes can provide an effective anonymity protection, even if adversaries are able to track the reputation transfer between pseudonyms for long periods of time. We have shown that the cost of a high degree of anonymity protection is the loss of reputation. If we assume that all clients adopt the same cloaking scheme, all clients will be similarly affected by the incurred loss in reputation. This is consistent with the typical use of reputation in participatory sensing applications, where the application server is mainly interested in comparing the reputation scores of different clients and correspondingly associate a weight to their sensor readings. In the rare case where the absolute reputation score is of interest, users may consider using a scheme, which preserves the reputation value but is more susceptible to attacks such as *Full* or *Floor*. However, if minor perturbations in the scores are acceptable, then schemes such as *RandScore*, *RandSet*, and *Hybrid* may be considered.

Table 10: Probability distribution for permanent reputation scores

		NEXT SCORE				
		-10	- 5	0	5	10
CURRENT SCORE	-10	0.35	0.30	0.25	0.08	0.02
	-5	0.25	0.30	0.30	0.10	0.05
	0	0.05	0.10	0.25	0.35	0.25
	5	0.05	0.10	0.30	0.30	0.25
	10	0.02	0.08	0.25	0.30	0.35

#### 7.4.2 Permanent Reputation Scores

Next, we assume permanent reputation scores, i.e., the reputation scores have an infinite period of validity. As such, in contrast with the previous scenario, the reputation cloaking schemes do not result in a loss of reputation at the end of each period, since the cloaked reputation can be stored for future use in subsequent time periods. In consequence, the adversaries need to consider the evolution of the reputation over several periods of time (instead of one period in the previous scenario). As the clients determine both the amount of transferred reputation and the time of the transfer, the number of possibilities for the reputation transfer increase. This in turn increases the complexity of the reputation analysis, thus making it harder for potential adversaries to de-anonymize users. In order to investigate the impact of permanent reputation scores on the anonymity protection, we adopt the following simulation setup and method. In particular, we highlight the differences with the previous evaluation detailed in Section 7.4.1.

##### Simulation Setup and Method

We investigate a population of 100 clients. Each simulation run covers 100 time intervals (if not noted otherwise) and we present the average results of 1000 simulations. During each time interval, the active clients generate five random sensor readings and report them to the application server. The application server attributes reputation scores to the pseudonyms according to the distribution presented in Table 10. Instead of arbitrarily attributing a random reputation score as in Section 7.4.1, the application server models possible behaviors of the clients by considering the last attributed reputation score in the computation of the new reputation score. The chosen distribution reflects our assumption that the participant behavior is unlikely to change at such short time scales. We thus assume that the probability that a client contributes a correct/incorrect sensor reading is higher when it has already contributed correct/incorrect samples, respectively. For example, assuming that a client obtains a reputation score of 10 for its last sensor reading, the probability that it obtains the same reputation score for its new sensor reading is equal to 0.35. In contrast, the probability that it obtains a reputation score of -10 is equal to 0.02. Note that we could apply the same distribution in the previous scenario, but the evolution of the reputation scores over time has a lower impact on anonymity protection than in this scenario due to the transient nature of the reputation scores.

Again, we assume that the RPM and the application server are malicious internal observers with the aim of identifying consecutive pseudonyms based on the information they have access to. The adversaries first identify the set of pseudonyms active in each time interval.  $S_{P_{\text{current}}}$  is the set of pseudonyms active in a given time interval and  $S_{P_{\text{next}}}$  is the set of pseudonyms active in the subsequent time interval. For each pseudonym  $P_{\text{current}}$ , the adversaries attempt to identify its successor  $P_{\text{next}}$  as follows.

The adversaries first identify the set of possible successors of  $P_{\text{current}}$  based on their reputation value and establish a temporary link between  $P_{\text{current}}$  and each candidate of  $S_{P_{\text{next}}}$  according to Algorithm 7. Next, the adversaries identify all pseudonyms in  $S_{P_{\text{current}}}$  that have a single link to a pseudonym in  $S_{P_{\text{next}}}$  and eliminate all other links, since a



**Algorithm 7** Identification of potential successors for permanent reputation scores

---

**Input:**  $R_{P_{current}}$ :  $P_{current}$ 's reputation,  $R_{P_{next}}$ :  $P_{next}$ 's reputation,  $S_{RT_{created}}$ : set of RTs created by  $P_{current}$ ,  $S_{RT_{hidden}}$ : set of RTs not used by  $P_{current}$ ,  $S_{RT_{used}}$ : set of RTs used by  $P_{next}$

- 1: **for all**  $P_{current} \in S_{P_{current}}$  **do**
- 2:   **for all**  $P_{next} \in S_{P_{next}}$  **do**
- 3:     **if**  $R_{P_{current}} + \text{Value}(R_{P_{hidden}}) \geq R_{P_{next}}$  **and**  $S_{RT_{used}} \subseteq (S_{RT_{created}} \cup S_{RT_{hidden}})$  **then**
- 4:       create a link between  $P_{current}$  and  $P_{next}$
- 5:     **end if**
- 6:   **end for**
- 7: **end for**

---

pseudonym has at most one successor. Similarly, they identify all pseudonyms in  $S_{P_{next}}$  having a single link to a pseudonym in  $S_{P_{current}}$  and eliminate all other links, as a pseudonym has at most one predecessor (see steps 4 to 8 in Algorithm 8). Note that we compute the number of potential successors  $x_k$  (i.e., links) a pseudonym  $k$  has on average after this step. For the remaining multiple links, the adversaries compute the probability of each link based on the probability that  $P_{current}$  handed RTs to the RPM and the probability that  $P_{next}$  shows a behavior similar to  $P_{current}$  (steps 16 and 23 in Algorithm 8). The probability that  $P_{next}$  shows a behavior similar to  $P_{current}$  is determined using the probability distribution presented in Table 10 (known by the application server which uses it for the computation of the reputation scores), while the probability that  $P_{current}$  handed RTs to the RPM is computed based on the probability  $p$  of using a RT, the number of transferred RTs  $k$ , and the total number of RTs  $n$ , as follows.

$$p(k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \quad (13)$$

For each  $P_{current}$  having multiple links, the probability of each link is normalized by the sum of the probability of all links. The link with the highest probability is then chosen as the potential successor of  $P_{current}$ . The other links are removed. The adversaries continue the reputation analysis by analyzing the remaining links according to the above steps (i.e., identification of single links and weighting of multiple links) until each pseudonym of  $S_{P_{current}}$  has at most one identified successor in  $S_{P_{next}}$ . At this stage, we calculate the length of the established pseudonym chains and the fraction of successors correctly identified by the adversaries in each time interval.

### Evaluation Results

We first investigate the impact of permanent reputation scores on the anonymity protection for a constant population of 100 clients based on the above setup and method. For this analysis, we consider that a reputation transfer is performed with probability  $p$  for each RT as in the *RandSet* scheme, except that the cloaked reputation is not lost at the end of the period but stored for future use by the clients. Figure 41 illustrates the influence of the choice of  $p$  (50%, 60%, 70%, 80%, 90%, 100%) on the fraction of potential successors over time. Note that  $p=100\%$  corresponds to a full transfer of reputation scores, i.e., no reputation cloaking is applied. The results show that applying reputation cloaking significantly improves anonymity protection. In particular, the degree of anonymity protection increases with decreasing  $p$ . For  $p=100\%$ , the percentage of potential successors is under 2% from  $t=17$ , whereas it lies between 49% and 90% for  $p$  equal to 90% and 50%, respectively. In comparison, using transient reputation scores, the percentage of potential successors is between 7% and 26% for the same values of  $p$  (see Figure 36b). The degree of anonymity protection thus considerably increases when using permanent reputation scores. This is due to the reputation scores stored by the clients, which individual value is unknown at both the RPM and the application server. Figure 43 highlights the impact of the stored reputation on the distribution of the reputation scores.

**Algorithm 8** Elimination of improbable successors for permanent reputation scores

---

**Input:**  $S_{\text{Links}}$ : set of links created in Algorithm 7

```

1: while  $\exists$  non-confirmed links  $\in S_{\text{Links}}$  do
2:   linkConfirmed  $\leftarrow$  false
3:   for all non-flagged pseudonym  $\in (S_{P_{\text{current}}} \cup S_{P_{\text{next}}})$  do
4:     if  $\exists$  a single link between  $P_{\text{current}}$  and  $P_{\text{next}}$  then
5:       remove all other links of  $P_{\text{current}}$  and  $P_{\text{next}}$ 
6:       flag  $P_{\text{current}}$  and  $P_{\text{next}}$ 
7:       update  $P_{\text{next}}$ 's hidden reputation
8:       linkConfirmed  $\leftarrow$  true
9:     end if
10:  end for
11:  if linkConfirmed = false then
12:    search for other non-flagged  $P_{\text{current}}$ s having the same potential successors
13:    if number of found  $P_{\text{current}}$ s = number of common  $P_{\text{next}}$ s then
14:      remove all other links of the found  $P_{\text{current}}$ s and  $P_{\text{next}}$ s
15:      flag the found  $P_{\text{current}}$ s and  $P_{\text{next}}$ s
16:      select for each  $P_{\text{current}}$  one of the found  $P_{\text{next}}$ s based on probability weighting
17:      update  $P_{\text{next}}$ 's hidden reputation
18:      linkConfirmed  $\leftarrow$  true
19:      break search
20:    end if
21:  end if
22:  if linkConfirmed = false then
23:    select the most likely link based on probability weighting
24:    remove all other links of the corresponding  $P_{\text{current}}$  and  $P_{\text{next}}$ 
25:    flag  $P_{\text{current}}$  and  $P_{\text{next}}$ 
26:    update the hidden reputation of  $P_{\text{next}}$ 
27:    linkConfirmed  $\leftarrow$  true
28:  end if
29: end while

```

---

Figure 43a presents the distribution of the reputation scores handed to the RPM to credit the reputation accounts and the reputation scores attributed by the application server. In comparison, Figure 43b shows the distribution of all reputation scores present in the system, i.e., the reputation scores stored by the clients for future use and those illustrated in Figure 43a. While adversaries know the total value of stored reputation scores, they however need to infer the individual value stored by each client each time a successor is selected according to the above steps. Figure 42 illustrates the success rate of the adversaries in inferring the value of the stored reputation, and hence the correctly identified successors. In absence of reputation cloaking, adversaries are able to correctly identify up to 60% of potential successors and to establish very long chains of unique predecessor-successor relations (up to a length of 92 in few cases as shown in Figure 44). In contrast, applying reputation cloaking reduces the fraction of correctly identified successors to 7% and 14% for  $p$  equal to 90% and 50%, respectively. Additionally, it leads to the construction of fewer and shorter pseudonym chains (up to a length of 75 for  $p=90\%$ ). We, however, observe that 99.9% of the pseudonyms are involved in chains of length of at most four pseudonyms. Reputation cloaking therefore effectively protects the anonymity of the clients, as the adversaries are able to link most of the pseudonyms for only short periods of time. Note that the chain lengths obtained for  $p=80\%$  cannot be directly compared to those obtained for the *RandSet* scheme (see Figure 40), as they are computed once the adversaries have weighted and selected one of the multiple links—a step not included in Section 7.4.1. In summary, the degree of anonymity protection is increased by both the utilization of permanent reputation scores and the application of reputation cloaking. Additionally, reputation cloaking does not incur permanent loss in reputation as compared to



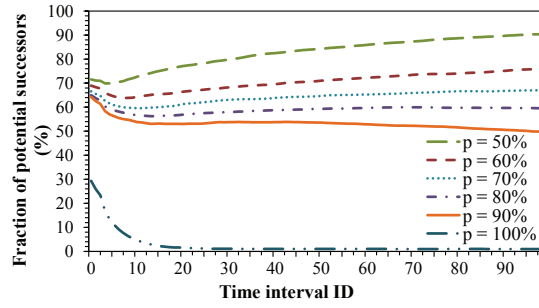


Figure 41: Fraction of potential successors over time for permanent reputation scores and a constant population of 100 clients using the *RandSet* scheme with different transfer probabilities  $p$

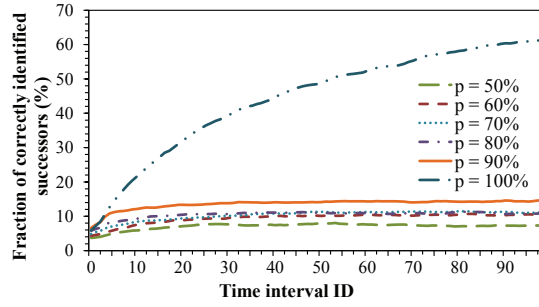
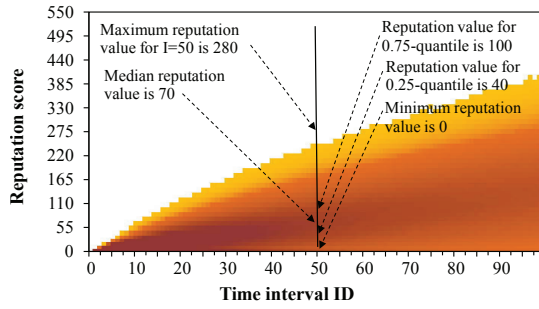
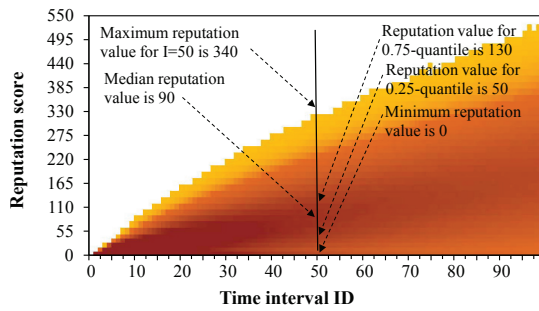


Figure 42: Fraction of correctly identified successors over time for permanent reputation scores and a constant population of 100 clients using the *RandSet* scheme with different transfer probabilities  $p$



(a) Transferred and attributed reputation scores



(b) Total reputation scores

Figure 43: Distribution of permanent reputation scores over time amongst clients for the *RandSet* scheme with  $p=80\%$  and for a constant population of 100 clients

our transient scheme, since the clients can store the cloaked reputation for future use. However, the transferred reputation still remains lower.

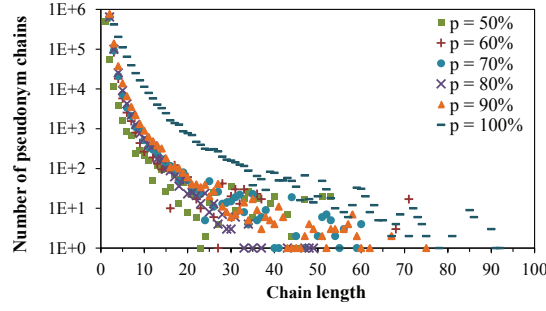


Figure 44: Number of identified pseudonym chains for permanent reputation scores and a constant population of 100 clients using the *RandSet* scheme

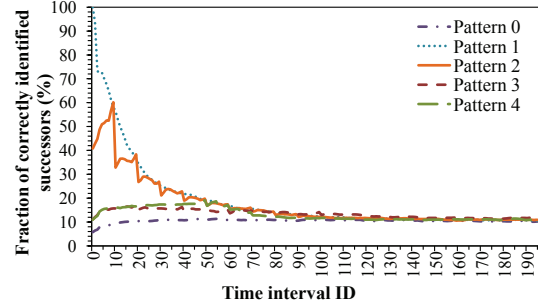


Figure 45: Fraction of correctly identified successors over time for permanent reputation scores and a variable population of up to 100 clients showing different activity patterns and using the *RandSet* scheme with  $p=80\%$

As compared to the transient scheme addressed in Section 7.4.1, we additionally consider a variable population of clients, i.e., clients become active after the beginning of the simulation. We use a constant population of 100 active clients as baseline for our analysis (*Pattern 0*) and selected four different activity patterns. Each activity pattern is defined by (a) the number of clients active at the simulation start, (b) start of the first period, (c) duration of the period between newly activated clients, and (d) number of activated clients after each period. For comparison purposes, we assume that the total number of clients is the same for all scenarios and that they become active latest at the 100th time interval.

- ▷ **PATTERN 0:** All 100 clients start at  $t=0$  s (reference scenario).
- ▷ **PATTERN 1:** A single client is active at  $t = 0$  and a new client becomes active at each time interval until reaching 100 active clients.
- ▷ **PATTERN 2:** Ten clients are active at the simulation start and ten new client become active every ten time intervals.
- ▷ **PATTERN 3:** 50 initial active clients and ten new clients every 20 time intervals.
- ▷ **PATTERN 4:** 50 clients are active at the beginning of the simulation and the next ten clients become active at  $t = 50$ . Subsequently, additional ten clients are introduced every five time intervals until 100 clients are active at the same time.

We further assume that all clients transfer their reputation scores with a probability  $p$  equal to 80%. This implies that the results for *Pattern 0* are the same as those previously discussed for a constant population of clients and  $p=80\%$ . In order to observe the effects of the latest activated clients on the degree of anonymity protection, we run each simulation for 200 time intervals (instead of 100 time intervals).

Figure 45 illustrates the impact of the introduction of new clients on the fraction of correctly identified successors for each activity pattern. It shows that introducing clients according to

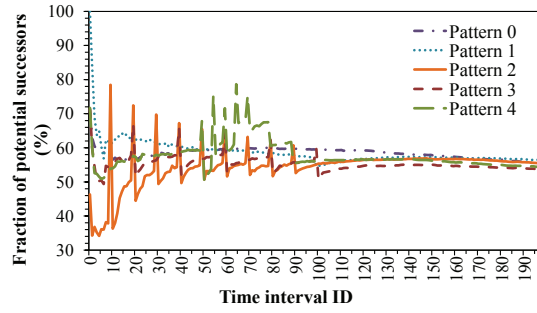


Figure 46: Fraction of potential successors over time for permanent reputation scores and a variable population of up to 100 clients showing different activity patterns and using the *RandSet* scheme with  $p=80\%$

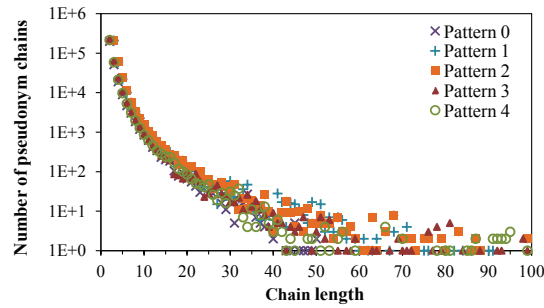


Figure 47: Number of identified pseudonym chains for permanent reputation scores and a variable population of up to 100 clients showing different activity patterns and using the *RandSet* scheme with  $p=80\%$

patterns 3 and 4 increases the number of correctly identified successors of up to 7% ( $t=20$ ) and 8% ( $t=50$ ) as compared to a constant population of clients (i.e., pattern 0), respectively. In comparison, patterns 1 and 2 lead to the correct identification of a higher number of successors. This difference is due to the number of introduced clients. Since new clients present similar reputation scores, the larger the group of new clients, the higher the number of potential successors for these clients and hence, the better the reciprocal protection between clients. Moreover, the difference between pattern 2 (i.e., ten clients introduced every ten intervals) and 3 (i.e., 20 clients introduced every ten intervals) shows that the introduction of ten more clients at each period significantly improves anonymity protection, since a lower number of successors can be identified by potential adversaries. Note that we consider the determination of the exact value of clients to introduce in order to guarantee anonymity protection as future work. Furthermore, pattern 1 reaches 100% of correctly identified successors in the first interval because there is only one client introduced at this time. Subsequently, the difficulty to correctly identify successors increases at each introduction of a new client. After the last introduction of new clients, the values of patterns 1 to 4 progressively stabilize around the value of pattern 0.

These results are mirrored in Figure 46. In particular, it shows for patterns 3 and 4 the augmentation of the number of potential successors caused by each introduction of clients. Note that pattern 4 shows a similar behavior to pattern 0 until the 50th interval as the population of clients remains constant. However, the degree of anonymity protection of the clients in pattern 4 is lower than those in pattern 0 during this interval, since the client population in pattern 4 is half the population of pattern 0. For all patterns, the variations in number of potential successors caused by new clients are temporary, as new clients progressively gain reputation scores. They hence become indistinguishable from clients which have been active for a longer period. This is illustrated in Figure 48, where the number of occurrences of low reputation scores temporarily augments at each introduction of new clients and later stabilizes. Since new clients have low reputation scores, they become potential

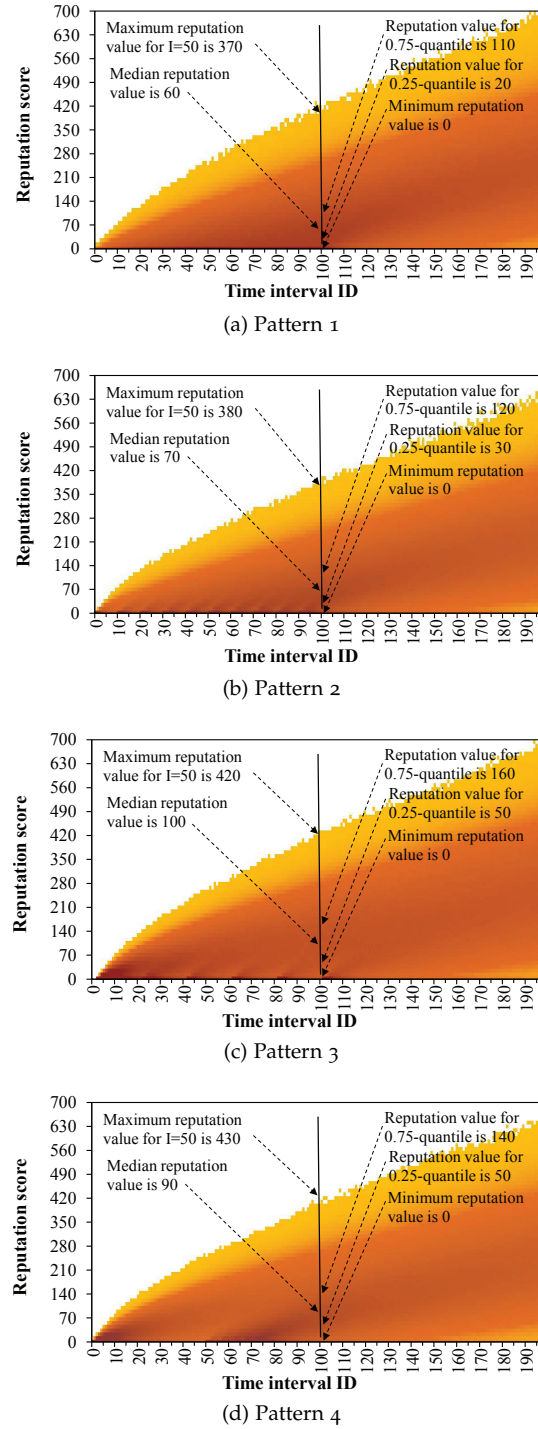


Figure 48: Distribution of reputation score amongst clients over time for permanent reputation scores and a variable population of up to 100 clients showing different activity patterns and using the *RandSet* scheme with  $p=80\%$

successors of already active clients. While the weight associated to the links between new and already active clients is low at the beginning, it progressively increases as the new clients gain reputation as shown in Figure 48. The length of successfully identified pseudonym chains increases with the introduction of new clients as illustrated in Figure 47. However, no difference can be clearly identified between patterns 1 to 4. Hence, no concrete conclusion can be drawn on the effect of different introduction patterns on the chain lengths. In summary,

Table 11: Measured overhead per RT and key generation for the clients

	RT	KEY PAIR
Average battery lifetime (hours)	5:09	5:18
Average number of executions	5037500	16860
Average execution time (ms)	4	1129

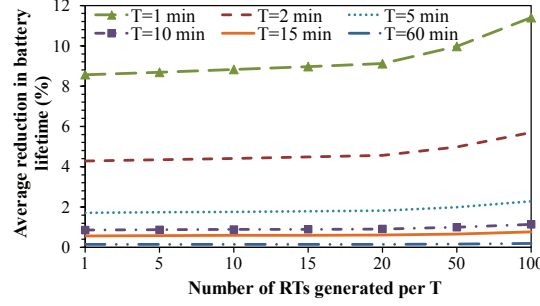


Figure 49: Estimated daily battery consumption incurred by the generation of pseudonyms and RTs

the results show that the degree of anonymity protection for a variable population of clients is lower than for a constant population. However, the anonymity protection increases with the number of simultaneously introduced clients, as they protect each other by becoming potential successors.

## 7.5 EMPIRICAL EVALUATION

We implemented a proof-of-concept of IncogniSense to demonstrate the feasibility of our approach for transient reputation scores. In particular, we quantified the overhead in terms of energy consumption for the clients. The overhead should be maintained as low as possible so as to not drain the battery of the mobile phones. In our implementation, the client program was developed on Android Nexus S phones, while the application server and RPM were implemented as two Apache Tomcat servlets. The activities of the clients are managed by a background thread. The blind RSA signatures are based on 1024-bit private/public key pairs. The pseudonyms and RTs are stored in a SQLite database on the clients, the access to which is strictly restricted to our application. The RPM especially maintains lists of the generated pseudonyms and utilized RTs in MySQL databases. It also controls the common time intervals and runs a synchronization function that determines the time drift between the server and the mobile devices. The clients, the RPM, and the application server communicate via Wi-Fi and the communications are secured using HTTPS.

We conducted a first experiment to measure the impact of the generation of pseudonyms, RTs, and keys on the clients' battery lifetime. We configured a benchmarking client to repeatedly execute the aforementioned sequence of operations until the exhaustion of the battery. We disabled all other programs and repeated each experiment 20 times on different clients. The results presented in Table 11 indicate that the overhead to generate keys for the new pseudonyms is significantly higher than the overhead to create RTs. Note that the overhead to create a pseudonym is equal to the overheads caused by both RT and key pair generation. IncogniSense saves 90% of energy while reducing the probability of reputation corruption from 0.1 to 0.0 as compared to [MR07] assuming  $n = 10$ . In a second experiment, we configured a client to generate pseudonyms and RTs with a period  $T$ . We examined the impact of both the duration of  $T$  and the number of generated RTs on the battery usage. Note that the impact of the cloaking schemes on the battery usage is negligible as compared to the cryptographic operations required to generate RTs and pseudonyms. Figure 49 shows the

overall reduction in battery lifetime per day imputed to our approach. By selecting  $T$  greater than 5 minutes, the daily cost in battery lifetime remains below 2.3%, rendering our approach especially feasible for energy-constrained devices such as mobile phones.

For permanent reputation scores, the blind RSA signature scheme prevents clients from lowering the value of the RTs without the knowledge of the RPM when applying the *RandScore* scheme. Another blind signature scheme needs hence to be applied. For example, the scheme proposed in [Ok95] is based on the principles of divisible electronic cash and enables to split the value of a RT into different *coins*. Each coin can then be handed over to the RPM independently of the others. The coins cannot be linked to the identity of the user, but to the originating RT. Therefore, the RPM obtains additional insights about the RTs that may improve its capabilities in linking consecutive pseudonyms.

## 7.6 RELATED WORK

We first compare IncogniSense with existing anonymity-preserving reputation systems designed for application domains orthogonal to participatory sensing. In fact, several reputation architectures based on pseudonyms have been proposed for peer-to-peer networks. For example, in [KP03], pseudonyms are created by Trusted Platform Modules and attested by Certificate Authorities. In the solution presented in [ACBMo8b], the users utilize a set of pseudonyms with the same reputation to collect electronic coins centralized at a third party. In contrast to our solution, both approaches are vulnerable to identity-based attacks since users can control an arbitrary number of pseudonyms. Another scheme presented in [MRM11] proposes context-based, self-certified, and Sybil-free pseudonyms built using e-tokens. The users, however, cannot change their pseudonyms once they are associated to a context. Finally, STARS [ZLK10] extends existing peer-to-peer reputation systems by traceability and anonymity. Complete anonymity of honest clients is however not guaranteed since the scheme reveals the identity of clients suspected to corrupt their reputation and is subject to false positives. In addition to peer-to-peer networks, the scheme introduced in [PSo8, Steog] unlinks the activities of users active in different Internet communities, while supporting cross-community reputation systems. As compared to our scheme, the focus is hence on the unlinkability of the same user across several applications (instead of one in our case) through the utilization of a identity management system. Moreover, the reputation scores are attributed by the users, instead of a central entity in our scheme.

IncogniSense shares the most similarities with [MR07] and [WH09], which rely on periodic pseudonyms and transfer of reputation between pseudonyms. As our work builds upon the RuP algorithm [MR07], we have presented a detailed comparison in Section 7.2.3. Recall that RuP requires the clients to create  $n$  temporary pseudonyms for each valid pseudonym and relies on probabilistic proofs. As such, a malicious client can create multiple pseudonyms and tamper with its reputation with a success probability of  $1/n$ . Increasing  $n$  reduces the threats to reputation manipulation, but increases associated overheads. In comparison to RuP, IncogniSense is robust against reputation corruption and the clients generate only one pseudonym per period. The protocol presented in [WH09], which extends the ideas from [MR07], is based on the concept of  $k$ -anonymity [Swe02]. They assume the existence of a third party server which forms groups of clients sharing the same reputation and distributes a signature key per group to sign the pseudonyms. The clients must however trust each other not to collude with the third party and/or other clients to reduce the anonymity set. Moreover, the proposed protocol introduces additional overhead for the clients to create pseudonyms as compared to [MR07] and our solution.

In the particular context of participatory sensing, [HKH12] makes the same assumptions as in IncogniSense, i.e., the utilization of periodic pseudonyms and transfer of reputations between pseudonyms. Instead of relying on cryptographic primitives, the proposed scheme is based on a trusted third party that generates equivalence classes based on the reputation values of the clients (cf. Section 3.3.2). In other words, this party dynamically creates groups of clients sharing the same reputation. In comparison to our scheme, the dynamic nature of

the proposed scheme optimizes the reduction in reputation. However, the trusted third party knows the mapping between the pseudonyms and the unique identity of the clients as well as the mapping between original and cloaked values. Consequently, users need to trust it not to reveal this information to other parties.

## 7.7 SUMMARY

In this chapter, we have presented our anonymity-preserving reputation framework called IncogniSense. Our framework is agnostic to both the applications and the applied reputation algorithm. It utilizes periodic pseudonyms which are generated using blind signature and relies on a secure reputation transfer mechanism between these pseudonyms. We have introduced the concept of reputation cloaking in order to prevent malicious application administrators from de-anonymizing the users by linking the reputation scores associated with their contributions. As cloaking has an inherent trade-off between anonymity protection and loss in reputation, we have explored the design space and extensively examined the performance of several different schemes. We have especially studied their resilience against linking attacks under various scenarios including the utilization of transient vs. permanent reputation scores and constant vs. variable client populations. Based on this analysis, we have provided guidelines for the choice of the appropriate cloaking scheme in accordance with the application requirements and the preferences of the users. We have further conducted a thorough threat analysis and demonstrated the robustness of IncogniSense against reputation corruption. We have finally evaluated the feasibility of our approach by implementing a proof-of-concept and measuring the incurred overhead in terms of energy consumption. The results prove that the additional energy expenditure incurred by our approach is minimal. It is hence a small price to pay for the enhanced privacy protection offered. To conclude, IncogniSense protects users against malicious application administrators who aim at breaching their privacy, while enabling the utilization of a reputation system. Moreover, it allows users to select the applied cloaking scheme according to their preferences in terms of privacy protection and reduction in reputation.





## CONCLUSIONS AND OUTLOOK

---

Study the past if you would divine the future.

Confucius

### 8.1 SUMMARY AND CONCLUSIONS

Encouraging contribution in participatory sensing applications and gaining widespread acceptance among participants requires multiple challenges to be addressed, especially the protection of users' privacy. Within the scope of this thesis, we have contributed to tackle this challenge by providing privacy-preserving schemes that follow two key directions. According to our first research direction, we have presented approaches that are customizable by the users according to their preferences. While existing solutions mainly focus on controlling the collection and the release of data, we have gone one step beyond by involving users in additional privacy decisions. For example, they can balance the tradeoff between loss in reputation and privacy protection according to their individual conception. Moreover, following our second research direction, we have reduced their dependency on central entities and developed schemes that protect user privacy despite the presence of malicious application administrators and third parties.

In particular, we have presented an collaborative and decentralized approach to preserve the location privacy of the users. Our application-agnostic approach is based on the physical exchange of collected sensor readings between users at opportunistic encounters. By doing so, users jumble parts of their paths with other users at each exchange. We have validated our approach based on a real-world dataset by measuring the impact of different exchange strategies on the system parameters, including exchange ratios, as well as the distance between original and jumbled paths. By running different outlier detection algorithms on realistic jumbled paths, we have shown that almost no jumbled sensor readings can be identified by potential malicious application administrators. Based on our evaluation, we have provided guidelines for the selection of exchange and reporting strategies to apply based on the desired privacy guarantees, the degree of trust in other users, and the willingness to equally share the reporting overhead. Using our approach, the users can therefore configure the applied exchange and reporting strategies depending on their preferences and directly control the protection of their privacy.

Our path jumbling approach, however, relies on the collaboration of the participants in the protection process. We have therefore proposed a scheme that assesses the trustworthy behavior of the users contributing to the aforementioned approach. It computes and attributes a trust level to each user based on statistics provided by the users about past exchanges. Based on the attributed trust levels, malicious users that drop exchanged sensor readings or inject incorrect sensor readings on purpose can be identified and subsequently excluded from the data jumbling process. Despite the multi-hop nature of our scheme, our extensive evaluations have shown that our approach performs well in identifying droppers and spammers in a realistic setup. After only few exchanges and reports, the identification of droppers significantly contributes to reduction in latency due to the exclusion of these users in further exchanges. Ultimately, users can assess the trust level of encountered users with our approach and leverage it to select the exchange and reporting strategies to apply according to their personal preferences.

We have finally proposed an anonymity-preserving reputation framework that addresses the inherent conflict between privacy and reputation. Our application-agnostic framework is based on periodic pseudonyms and a secure reputation transfer between these pseudonyms. We

have introduced the concept of reputation cloaking in order to prevent malicious application administrators from de-anonymizing users by linking the reputation scores associated with their contributions. As cloaking shows an inherent trade-off between anonymity protection and loss in reputation, we have explored the design space and extensively examined the performance of several different schemes. We have especially studied their resilience against linking attacks under various scenarios. Based on this analysis, we have provided guidelines for choosing the appropriate cloaking scheme based on the preferences of the users in terms of loss in reputation and privacy protection. We have demonstrated the robustness of our framework against reputation corruption. We have validated our approach by a proof-of-concept implementation and shown that the additional energy expenditure incurred by our approach is minimal.

This thesis presents major contributions towards the protection of privacy in participatory sensing applications. Our solutions are controllable and configurable by the users in order to reflect their personal and individual preferences. By involving users in the proposed mechanisms and making them transparent, we strive for increasing the users' trust in participatory sensing campaigns and subsequently, foster their contribution to such applications. We have designed all contributions assuming that application administrators and related entities can be malicious and especially willing to breach the privacy of contributing participants. By reducing the users' dependency on these entities, we cater for solutions that improve their privacy protection under such realistic scenarios. In conclusion, we have proposed, analyzed, and validated novel user-controlled privacy-preserving approaches that are independent on the integrity of the application administrators and involved third parties. This leads us to expect a widespread improvement of the privacy protection under real-world conditions.

## 8.2 OUTLOOK

The contributions of this thesis lay the foundations for user-controlled privacy-preserving solutions specially tailored to participatory sensing. Based on their design and realization, different new research questions and challenges have come to light:

- ▷ **USABLE AND COMPREHENSIVE INTERACTIONS** We have herein proposed technical solutions that can be configured and controlled by the users according to their preferences. However, providing appropriate user interfaces as support for this control is still a challenge to be tackled. The utilization of small form factor mobile phones combined with the complexity of the settings' implications on the users' privacy makes the design of such interfaces very challenging. Indeed, users should not only be able to simply select any settings, but also understand their choices and their potential consequences in a user-friendly and usable way. This includes limiting the number of required interactions to the minimum in order to reduce the users' burdens while giving enough freedom to the users to personalize the schemes. Tackling this challenge requires the evaluation of the designed solutions by users of participatory sensing applications in order to explore their preferences.
- ▷ **LARGE SCALE REAL-WORLD DEPLOYMENTS** In addition to evaluate the appropriateness of such interactions, long-term user studies would provide valuable insights about how users make use of privacy-preserving solutions in the field. This includes investigating if and how they change their privacy settings or which are the working solutions and parameter sets. Outcomes of such studies could allow to refine the design of the proposed solutions and tailor them to the real needs of users having tested them under real-world conditions. A real-world deployment would also enable to gather insights from a technical perspective about the performance of the developed schemes and fine-tune their design to best fit the experienced conditions.
- ▷ **PRIVACY AWARENESS** Beyond the approaches proposed in this thesis, another challenge to tackle is to increase the awareness of potential privacy threats resulting from the

utilization of mobile applications. While the public awareness has incrementally raised in recent years, it still does not reach the same degree as in other domains, such as the recording of license plates on cars and the utilization of Closed-Circuit Television (CCTV) cameras by the police. Surveillance by the governmental entities is often perceived as a threat to the privacy of the citizens. In contrast, participatory sensing applications may appear more inoffensive since they are not controlled by central and official bodies. However, the development of new applications is open to anybody and the control over these application is limited to the verifications conducted by the App stores before the applications are released. Hence, innovative methods still need to be invented to efficiently increase the awareness of the users about these particular privacy issues. Existing solutions, such as textual warnings, may be insufficient to fulfill this goal, since they are often discarded by the users without having been read.

- ▷ **MULTI-PARTY PRIVACY PROTECTION** While it has been shown in [TLH<sup>+</sup>10] that participants value the location privacy of their friends, most of the current privacy-preserving mechanisms focus on the protection of the participants themselves. Yet the privacy of people in their surroundings may also be threatened. For example, faces of uninvolved people can appear in the contributed images as shown in DietSense [RPH<sup>+</sup>07] (cf. Section 2.1.1). In current systems, it is mostly the user's responsibility to protect the privacy of bystanders. However, malicious participants may also deliberately distribute compromising information about others. Automated solutions to minimize the captured data such that it does not violate the privacy of others remain to be developed.
- ▷ **IDENTITY MANAGEMENT** Existing privacy-aware solutions tailored to the requirements of mobile sensing applications often consider the identity and the location of the users as two distinct kinds of information to protect. In domains orthogonal to mobile sensing such as location-based services, location is, however, seen as part of the user's identity. By including additional information captured by onboard sensors, the concept of digital identity introduced in these domains could be extended and applied in mobile sensing applications. This would enable the investigation of new research questions related to identity management in participatory sensing and refine the definition of identity by including further privacy-relevant information, such as sensor data collected using mobile phones.

In conclusion, participatory sensing applications are about to cross the chasm to mass deployment, yet open challenges specially in the domain of privacy protection persist. Addressing these challenges would foster the contribution to participatory applications and benefit both the research community as well as the society at large.



## BIBLIOGRAPHY

---

- [AA01] D. Agrawal and C. Aggarwal: *On the Design and Quantification of Privacy Preserving Data Mining Algorithms*. In *Proceedings of the 12th ACM Symposium on Principles of Database Systems (PODS)*, pages 247–255, 2001.
- [AAB<sup>+</sup>07] T. F. Abdelzaher, Y. Anokwa, P. Boda, J. A. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich: *Mobiscopes for Human Spaces*. *IEEE Pervasive Computing*, 6(2):20–29, 2007.
- [ACBMo8b] E. Androulaki, S. G. Choi, S. M. Bellovin, and T. Malkin: *Reputation Systems for Anonymous Networks*. In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS)*, pages 202–218, 2008.
- [ACC<sup>+</sup>05] C. Andersson, J. Camenisch, S. Crane, S. Fischer-Hübner, R. Leenes, S. Pearsorr, J. Pettersson, and D. Sommer: *Trust in PRIME*. In *Proceedings of the 5th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 552–559, 2005.
- [ACC09] M. Azizyan, I. Constandache, and R. R. Choudhury: *SurroundSense: Mobile Phone Localization via Ambience Fingerprinting*. In *Proceedings of the 15th ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 261–272, 2009.
- [AG05] A. Acquisti and J. Grossklags: *Privacy and Rationality in Individual Decision Making*. *IEEE Security and Privacy*, 3(1):26–33, 2005.
- [AMCK<sup>+</sup>02] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. D. Mickunas, and S. Yi: *Routing through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments*. In *Proceedings of 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 74–83, 2002.
- [AML<sup>+</sup>10] G.-S. Ahn, M. Musolesi, H. Lu, R. Olfati-Saber, and A. T. Campbell: *MetroTrack: Predictive Tracking of Mobile Events Using Mobile Phones*. *Distributed Computing in Sensor Systems*, 6131(1):230–243, 2010.
- [AMM<sup>+</sup>08] M. Annavaram, N. Medvidovic, U. Mitra, S. Narayanan, G. Sukhatme, Z. Meng, S. Qiu, R. Kumar, G. Thatte, and D. Spruijt-Metz: *Multimodal Sensing for Pediatric Obesity Applications*. In *Proceedings of International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense)*, pages 21–25, 2008.
- [APN<sup>+</sup>12a] B. Agir, T. G. Papaioannou, R. Narendula, K. Aberer, and J.-P. Hubaux: *Adaptive Personalized Privacy in Participatory Sensing*. Technical Report EPFL-REPORT-176115, Ecole Polytechnique Fédérale de Lausanne, 2012.
- [APN<sup>+</sup>12b] B. Agir, T. G. Papaioannou, R. Narendula, K. Aberer, and J.-P. Hubaux: *Adaptive Personalized Privacy in Participatory Sensing (Extended Abstract)*. In *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2012.
- [App13] Apple Inc.: *iTunes Store*. Online: <https://www.apple.com/itunes> (accessed in 01.2013), 2013.

- [AS00] R. Agrawal and R. Srikant: *Privacy-preserving Data Mining*. ACM Sigmod Record, 29(2):439–450, 2000.
- [AS11] C. Arthur and K. Stuart: *PlayStation Network Users Fear Identity Theft after Major Data Leak*. Online: <http://www.guardian.co.uk/technology/2011/apr/27/playstation-users-identity-theft-data-leak> (accessed in 01.2013), 2011.
- [BC10] X. Bao and R. R. Choudhury: *MoVi: Mobile Phone based Video Highlights via Collaborative Sensing*. In *Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 357–370, 2010.
- [BCPo8] C. Boldrini, M. Conti, and A. Passarella: *Exploiting Users' Social Relations to Forward Data in Opportunistic Networks: The HiBOp Solution*. *Pervasive and Mobile Computing*, 4(5):633–657, 2008.
- [BEH<sup>+</sup>06] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava: *Participatory Sensing*. In *Proceedings of the 1st Workshop on World-Sensor-Web (WSW)*, pages 1–5, 2006.
- [Bero5] A. R. Beresford: *Location Privacy in Ubiquitous Computing*. Technical Report 612, University of Cambridge, 2005. <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-612.pdf>.
- [BFV12] L. Becchetti, L. Filipponi, and A. Vitaletti: *Privacy Support in People-centric Sensing*. *Journal of Communications*, 7(8):606–621, 2012.
- [BKNS00] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander: *LOF: Identifying Density-Based Local Outliers*. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 93–104, 2000.
- [Bru78] P. Brucker: *On the Complexity of Clustering Problems*. *System Modeling and Optimization*, 157(1):45–54, 1978.
- [BS03] A. Beresford and F. Stajano: *Location Privacy in Pervasive Computing*. *IEEE Pervasive Computing*, 2(1):46–55, 2003.
- [BW90] L. Brandeis and S. Warren: *The Right to Privacy*. *Harvard Law Review*, 4(5):193–220, 1890.
- [CBC09] CBC News: *Depressed Woman Loses Benefits over Facebook Photos*. Online: <http://www.cbc.ca/news/canada/montreal/story/2009/11/19/quebec-facebook-sick-leave-benefits.html> (accessed in 01.2013), 2009.
- [CBCC10] I.-R. Chen, F. Bao, M. Chang, and J.-H. Cho: *Trust Management for Encounter-Based Routing in Delay Tolerant Networks*. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, 2010.
- [CBS11] CBS News: *Did the Internet Kill Privacy? Facebook Photos Lead to a Teacher Losing her Job; What Expectations of Privacy Exist in the Digital Era?* Online: <http://www.cbsnews.com/stories/2011/02/06/sunday/main7323148.shtml> (accessed in 01.2013), 2011.
- [CCCS11] H. Choi, S. Chakraborty, Z. Charbiwala, and M. B. Srivastava: *SensorSafe: A Framework for Privacy-Preserving Management of Personal Sensory Information*. In *Proceedings of the 8th VLDB International Conference on Secure Data Management (SDM)*, pages 85–100, 2011.
- [CCL<sup>+</sup>09] R. Cáceres, L. Cox, H. Lim, A. Shakimov, and A. Varshavsky: *Virtual Individual Servers as Privacy-preserving Proxies for Mobile Devices*. In *Proceedings of the 1st ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds (MobiHeld)*, pages 37–42, 2009.

- [CCS12] H. Choi, S. Chakraborty, and M. B. Srivastava: *Design and Evaluation of Sensor-Safe: A Framework for Achieving Behavioral Privacy in Sharing Personal Sensory Information*. In *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1004–1011, 2012.
- [CEL<sup>+</sup>06] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson: *People-centric Urban Sensing*. In *Proceedings of the 2nd Annual International Wireless Internet Conference (WICON)*, pages 18–31, 2006.
- [CEL<sup>+</sup>08] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn: *The Rise of People-centric Sensing*. *IEEE Internet Computing*, 12(4):12–21, 2008.
- [CGR<sup>+</sup>11] D. Christin, J. Guillemet, A. Reinhardt, M. Hollick, and S. S. Kanhere: *Privacy-preserving Collaborative Path Hiding for Participatory Sensing Applications*. In *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 341–350, 2011.
- [CH11] D. Christin and M. Hollick: *We Must Move – We Will Move: On Mobile Phones as Sensing Platforms*. In *Proceedings of the 10th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 25–28, 2011.
- [Cha81] D. Chaum: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*. *Communications of the ACM*, 24(2):84–90, 1981.
- [Cha83] D. Chaum: *Blind Signatures for Untraceable Payments*. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203, 1983.
- [CRH<sup>+</sup>12a] D. Christin, C. Roßkopf, M. Hollick, L. A. Martucci, and S. S. Kanhere: *Incog-niSense: An Anonymity-preserving Reputation Framework for Participatory Sensing Applications*. In *Proceedings of the 10th IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 135–143, 2012.
- [CRH<sup>+</sup>13] D. Christin, C. Roßkopf, M. Hollick, L. A. Martucci, and S. S. Kanhere: *Incog-niSense: An Anonymity-preserving Reputation Framework for Participatory Sensing Applications*. Accepted for publication in *Pervasive and Mobile Computing*, 2013.
- [CRKH11] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick: *A Survey on Privacy in Mobile Participatory Sensing Applications*. *Journal of Systems and Software*, 84(11):1928–1946, 2011.
- [CRPSKH12a] D. Christin, D. Rodriguez Pons-Sorolla, S. S. Kanhere, and M. Hollick: *Trust-Meter: A Trust Assessment Framework for Collaborative Path Hiding in Participatory Sensing Applications*. Technical Report TR-SEEMOO-2012-02, Technische Universität Darmstadt, 2012.
- [CS97] J. Camenisch and M. Stadler: *Efficient Group Signature Schemes for Large Groups*. *Advances in Cryptology (CRYPTO)*, 1294:410–424, 1997.
- [CsS<sup>+</sup>05] J. Camenisch, a. shelat, D. Sommer, S. Fischer-Hübner, M. Hansen, H. Krase-mann, G. Lacoste, R. Leenes, and J. Tseng: *Privacy and Identity Management for Everyone*. In *Proceedings of the 1st ACM Workshop on Digital Identity Management (DIM)*, pages 20–27, 2005.
- [CYCS10] J. Carrapetta, N. Youdale, A. Chow, and V. Sivaraman: *Haze Watch Project*. Online: <http://www.pollution.ee.unsw.edu.au> (accessed in 01.2013), 2010.

- [CYHEo6] K. Chang, N. Yau, M. Hansen, and D. Estrin: *SensorBase.org - A Centralized Repository to Slog Sensor Network Data*. In *Proceedings of the Euro-American Workshop on Middleware for Sensor Networks (EAWMS)*, pages 116–128, 2006.
- [DAC<sup>+</sup>09] T. Denning, A. Andrew, R. Chaudhri, C. Hartung, J. Lester, G. Borriello, and G. Duncan: *BALANCE: Towards a Usable Pervasive Wellness Application with Accurate Activity Inference*. In *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 5:1–5:6, 2009.
- [DCo9] L. Deng and L. Cox: *LiveCompare: Grocery Bargain Hunting through Participatory Sensing*. In *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 1–6, 2009.
- [DCS11] E. De Cristofaro and C. Soriente: *Short Paper: PEPSI—Privacy-Enhanced Participatory Sensing Infrastructure*. In *Proceedings of the 4th ACM Conference on Wireless Network Security (WiSec)*, pages 23–28, 2011.
- [DFMSo2] J. Domingo-Ferrer and J. Mateo-Sanz: *Practical Data-oriented Microaggregation for Statistical Disclosure Control*. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.
- [DKCBo8] Y. Dong, S. S. Kanhere, C. Chou, and N. Bulusu: *Automatic Collection of Fuel Prices from a Network of Mobile Cameras*. In *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 140–156, 2008.
- [DLHHo9] B. Debatin, J. P. Lovejoy, A.-K. Horn, and B. N. Hughes: *Facebook and Online Privacy: Attitudes, Behaviors, and Unintended Consequences*. *Journal of Computer-Mediated Communication*, 15(1):83–108, 2009.
- [DMP<sup>+</sup>10] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma: *PRISM: Platform for Remote Sensing using Smartphones*. In *Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 63–76, 2010.
- [DMSo4] R. Dingledine, N. Mathewson, and P. Syverson: *Tor: The Second-generation Onion Router*. In *Proceedings of the 13th Conference on USENIX Security Symposium (USENIX Security)*, pages 21–38, 2004.
- [DNBB10] M. O. Derawi, C. Nickel, P. Bours, and C. Busch: *Unobtrusive User-authentication on Mobile Phones using Biometric Gait*. In *Proceeding of the 6th IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 306–311, 2010.
- [Douo2] J. R. Douceur: *The Sybil Attack*. In *Revised Papers from the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 251–260, 2002.
- [DY83] D. Dolev and A. C. Yao: *On the Security of Public Key Protocols*. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
- [ECo6] S. B. Eisenman and A. T. Campbell: *SkiScape Sensing*. In *Proceedings of the 4th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 401–402, 2006.
- [EGSo3] A. Evfimievski, J. Gehrke, and R. Srikant: *Limiting Privacy Breaches in Privacy Preserving Data Mining*. In *Proceedings of the 22th ACM Symposium on Principles of Database Systems (PODS)*, pages 211–222, 2003.



- [ELCo8] S. B. Eisenman, N. D. Lane, and A. T. Campbell: *Techniques for Improving Opportunistic Sensor Networking Performance*. In *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 157–175, 2008.
- [ELM<sup>+</sup>06] S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, G. Ahn, and A. T. Campbell: *MetroSense Project: People-centric Sensing at Scale*. In *Proceedings of the 1st Workshop on World-Sensor-Web (WSW)*, pages 6–11, 2006.
- [EML<sup>+</sup>07] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell: *The BikeNet Mobile Sensing System for Cyclist Experience Mapping*. In *Proceedings of the 5th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 87–101, 2007.
- [EML<sup>+</sup>09] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell: *BikeNet: A Mobile Sensing System for Cyclist Experience Mapping*. *ACM Transactions on Sensor Networks*, 6(1):1–39, 2009.
- [Est10] D. Estrin: *Participatory Sensing: Applications and Architecture*. In *Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 3–4, 2010.
- [FHHK<sup>+</sup>11] S. Fischer-Hübner, C. Hoofnagle, I. Krontiris, K. Rannenberg, and M. Waidner: *Online Privacy: Towards Informational Self-Determination on the Internet (Dagstuhl Perspectives Workshop 11061)*. *Dagstuhl Manifestos*, 1(1):1–20, 2011.
- [FHP04] S. Fischer-Hübner and J. S. Pettersson: *Evaluation of Early Prototypes (D[6-12].1.a)*. Technical Report PRIME project deliverable Do6.1.b, 2004.
- [GDG<sup>+</sup>05] N. Good, R. Dhamija, J. Grossklags, D. Thaw, S. Aronowitz, D. Mulligan, and J. Konstan: *Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware*. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*, pages 43–52, 2005.
- [Geno8] R. Gennaro: *Digital Cash*. In S. Goldwasser and M. Bellare (editors): *Lecture Notes on Cryptography*, chapter 12.5, pages 233–237. 2008.
- [GFHo9] N. Györbíró, A. Fábián, and G. Hományi: *An Activity Recognition System for Mobile Phones*. *Mobile Networks and Applications*, 14(1):82–91, 2009.
- [GHW<sup>+</sup>11] C. Graf, C. Hochleitner, P. Wolkerstorfer, J. Angulo, S. Fischer-Hübner, and E. Wästlund: *Towards Usable Privacy Enhancing Technologies: Lessons Learned from the PrimeLife Project*. Technical Report PrimeLife Deliverable D4.1.6, 2011.
- [GKN<sup>+</sup>07] W. I. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao: *SenseWeb: An Infrastructure for Shared Sensing*. *IEEE Multimedia*, 14(4):8–13, 2007.
- [GLC<sup>+</sup>08] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt: *Micro-Blog: Sharing and Querying Content through Mobile Phones and Social Participation*. In *Proceedings of the 6th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 174–186, 2008.
- [Goo13] Google Inc.: *Google Play Store*. Online: <https://play.google.com> (accessed in 01.2013), 2013.
- [GPA<sup>+</sup>10] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher: *GreenGPS: A Participatory Sensing Fuel-efficient Maps Application*. In *Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 151–164, 2010.

- [GPTAo8] R. K. Ganti, N. Pham, Y. Tsai, and T. F. Abdelzaher: *PoolView: Stream Privacy for Grassroots Participatory Sensing*. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*, pages 281–294, 2008.
- [Gro12] D. Gross: *Yahoo Hacked, 450,000 Passwords Posted Online*. Online: <http://edition.cnn.com/2012/07/12/tech/web/yahoo-users-hacked/index.html> (accessed in 01.2013), 2012.
- [HBZ<sup>+</sup>o6] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden: *CarTel: A Distributed Mobile Sensor Computing System*. In *Proceedings of the 4th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 125–138, 2006.
- [HFHPBo7] M. Hansen, S. Fischer-Hübner, J. Pettersson, and M. Bergmann: *Transparency Tools for User-controlled Identity Management*. In *Proceedings of the 17th eChallenges Conference (e-2007)*, pages 1360–1367, 2007.
- [HG05] B. Hoh and M. Gruteser: *Protecting Location Privacy Through Path Confusion*. In *Proceedings of the 1st IEEE International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM)*, pages 194–205, 2005.
- [HGH<sup>+</sup>o8] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson: *Virtual Trip Lines for Distributed Privacy-preserving Traffic Monitoring*. In *Proceedings of the 6th ACM International Conference on Mobile systems, Applications, and Services (MobiSys)*, pages 15–28, 2008.
- [HIJ<sup>+</sup>12] B. Hoh, T. Iwuchukwu, Q. Jacobson, D. Work, A. M. Bayen, R. Herring, J.-C. Herrera, M. Gruteser, M. Annavaram, and J. Ban: *Enhancing Privacy and Accuracy in Probe Vehicle-Based Traffic Monitoring via Virtual Trip Lines*. *IEEE Transactions on Mobile Computing*, 11(5):849–864, 2012.
- [HKHo9] K. L. Huang, S. S. Kanhere, and W. Hu: *Towards Privacy-sensitive Participatory Sensing*. In *Proceedings of the 5th IEEE Workshop on Sensor Networks and Systems for Pervasive Computing (PerSENS)*, pages 1–6, 2009.
- [HKH10a] K. L. Huang, S. S. Kanhere, and W. Hu: *Are you Contributing Trustworthy Data?: The Case for a Reputation System in Participatory Sensing*. In *Proceedings of the 13th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWIM)*, pages 14–22, 2010.
- [HKH10b] K. L. Huang, S. S. Kanhere, and W. Hu: *Preserving Privacy in Participatory Sensing Systems*. *Computer Communications*, 33(11):1266–1280, 2010.
- [HKH12] K. L. Huang, S. S. Kanhere, and W. Hu: *A Privacy-preserving Reputation System for Participatory Sensing*. In *Proceedings of the 37th IEEE Conference on Local Computer Networks (LCN)*, pages 10–18, 2012.
- [Int11] International Communication Union: *The World in 2011: ICT Facts and Figures*. Online: <http://www.itu.int> (accessed in 01.2013), 2011.
- [KBRL09] E. Kanjo, J. Bacon, D. Roberts, and P. Landshoff: *MobSens: Making Smart Phones Smarter*. *IEEE Pervasive Computing*, 8(4):50–57, 2009.
- [KGZo7] A. Kansal, M. Goraczko, and F. Zhao: *Building a Sensor Network of Mobile Phones*. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 547–548, 2007.
- [KHKZo8] A. Krause, E. Horvitz, A. Kansal, and F. Zhao: *Toward Community Sensing*. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 481–492, 2008.

- [KKT09] A. Kapadia, D. Kotz, and N. Triandopoulos: *Opportunistic Sensing: Security Challenges for the New Paradigm*. In *Proceedings of the 1st International Conference on Communication Systems and Networks (COMNETS)*, pages 1–10, 2009.
- [KNT00] E. M. Knorr, R. T. Ng, and V. Tucakov: *Distance-Based Outliers: Algorithms and Applications*. *Very Large Data Bases Journal (VLDB)*, 8(3):237–253, 2000.
- [KOK09] A. Keränen, J. Ott, and T. Kärkkäinen: *The ONE Simulator for DTN Protocol Evaluation*. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools)*, pages 55:1–55:10, 2009.
- [KP03] M. Kinateder and S. Pearson: *A Privacy-Enhanced Peer-to-Peer Reputation System*. *E-Commerce and Web Technologies*, 2738:206–215, 2003.
- [Kru07] J. Krumm: *Inference Attacks on Location Tracks*. In *Proceedings of the 5th IEEE International Conference on Pervasive Computing (Pervasive)*, pages 127–143, 2007.
- [KS11a] L. Kazemi and C. Shahabi: *Towards Preserving Privacy in Participatory Sensing*. In *Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications (PERCOM Workshop)*, pages 328–331, 2011.
- [KS11b] L. Kazemi and C. Shahabi: *A Privacy-aware Framework for Participatory Sensing*. *ACM SIGKDD Explorations Newsletter*, 13(1):43–51, 2011.
- [KTH10] U. Kumar, G. Thakur, and A. Helmy: *PROTECT: Proximity-Based Trust-Advisor using Encounters for Mobile Societies*. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 636–645, 2010.
- [LCC<sup>+</sup>05] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit: *Place Lab: Device Positioning using Radio Beacons in the Wild*. *Pervasive Computing*, 3468(1):116–133, 2005.
- [Liu07] L. Liu: *From Data Privacy to Location Privacy: Models and Algorithms*. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 1429–1430, 2007.
- [LKBG06] L. Lilien, Z. H. Kamal, V. Bhuse, and A. Gupta: *Opportunistic Networks: The Concept and Research Challenges in Privacy and Security*. In *Proceedings of the International Workshop on Research Challenges in Security and Privacy for Mobile and Wireless Networks (WSPWN)*, pages 134–147, 2006.
- [LLECo8] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell: *Bubble-sensing: A New Paradigm for Binding a Sensing Task to the Physical World using Mobile Phones*. In *Proceedings of the International Workshop on Mobile Devices and Urban Sensing (MODUS)*, pages 58–71, 2008.
- [LLEC10] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell: *Bubble-sensing: Binding Sensing Tasks to the Physical World*. *Pervasive and Mobile Computing*, 6(1):58–71, 2010.
- [LPL<sup>+</sup>09] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell: *SoundSense: Scalable Sound Sensing for People-centric Applications on Mobile Phones*. In *Proceedings of the 7th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 165–178, 2009.
- [LS93] R. Lunheim and G. Sindre: *Privacy and Computing: a Cultural Perspective*. In *Proceedings of the IFIP TC9/WG9.6 Working Conference on Security and Control of Information Technology in Society*, pages 25–40, 1993.

- [MC09] J. Meyerowitz and R. R. Choudhury: *Hiding Stars with Fireworks: Location Privacy through Camouflage*. In *Proceedings of the 15th ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 345–356, 2009.
- [MC10] J. T. Meyerowitz and R. R. Choudhury: *CacheCloak: Enabling Real-time Location Privacy for Mobile Users*. *ACM SIGMOBILE Mobile Computing and Communications Review*, 13(3):38–41, 2010.
- [MFD03] G. Myles, A. Friday, and N. Davies: *Preserving Privacy in Environments with Location-based Applications*. *IEEE Pervasive Computing*, 2(1):56–64, 2003.
- [MHM<sup>+</sup>10] M. Mun, S. Hao, N. Mishra, K. Shilton, J. A. Burke, D. Estrin, M. Hansen, and R. Govindan: *Personal Data Vaults: A Locus of Control for Personal Data Streams*. In *Proceedings of 6th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 17:1–17:12, 2010.
- [Mic13] Microsoft Corporation: *GeoLife GPS Trajectories*. Online: <http://research.microsoft.com/en-us/projects/geolife> (accessed in 01.2013), 2013.
- [MKGVo7] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam: *L-diversity: Privacy beyond K-anonymity*. *ACM Transactions on Knowledge Discovery from Data*, 1(1):1–52, 2007.
- [MLF<sup>+</sup>08] E. Miluzzo, N. D. Lane, K. Fodor, R. A. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell: *Sensing meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application*. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*, pages 337–350, 2008.
- [MML<sup>+</sup>08] M. Musolesi, E. Miluzzo, N. D. Lane, S. B. Eisenman, T. Choudhury, and A. T. Campbell: *The Second Life of a Sensor: Integrating Real-world Experience in Virtual Worlds using Mobile Phones*. In *Proceedings of the 5th Workshop on Embedded Networked Sensors (HotEmNets)*, pages 1–5, 2008.
- [MPRo8] P. Mohan, V. Padmanabhan, and R. Ramjee: *Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones*. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*, pages 323–336, 2008.
- [MR07] H. Miranda and L. Rodrigues: *A Framework to Provide Anonymity in Reputation Systems*. In *Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MobiQuitous)*, pages 1–4, 2007.
- [MRM11] L. A. Martucci, S. Ries, and M. Mühlhäuser: *Sybil-Free Pseudonyms, Privacy and Trust: Identity Management in the Internet of Services*. *Journal of Information Processing*, 19(1):1–15, 2011.
- [MRS<sup>+</sup>09] M. Mun, S. Reddy, K. Shilton, N. Yau, J. A. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda: *PEIR, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research*. In *Proceedings of the 7th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 55–68, 2009.
- [MSC09] J. Manweiler, R. Scudellari, and L. P. Cox: *SMILE: Encounter-Based Trust for Mobile Social Services*. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, pages 246–255, 2009.
- [MSNS09] N. Maisonneuve, M. Stevens, M. E. Niessen, and L. Steels: *NoiseTube: Measuring and Mapping Noise Pollution with Mobile Phones*. In *Proceedings of the 4th International Symposium on Information Technologies in Environmental Engineering (ITEE)*, pages 215–228, 2009.

- [NBB<sup>+</sup>10] L. Nachman, A. Baxi, S. Bhattacharya, V. Darera, P. Deshpande, N. Kodalapura, V. Mageshkumar, S. Rath, J. Shahabdeen, and R. Acharya: *Jog Falls: A Pervasive Healthcare Platform for Diabetes Management*. *Pervasive Computing*, 6030(1):94–111, 2010.
- [New95] P. B. Newell: *Perspectives on Privacy*. *Journal of Environmental Psychology*, 15(2):87–104, 1995.
- [Niso04] H. Nissenbaum: *Privacy as Contextual Integrity*. *Washington Law Review*, 79(1):101–139, 2004.
- [OHD<sup>+</sup>12] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang: *ACcessory: Password Inference Using Accelerometers on Smartphones*. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 9:1–9:6, 2012.
- [Oka95] T. Okamoto: *An Efficient Divisible Electronic Cash Scheme*. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 438–451, 1995.
- [PFHD<sup>+</sup>05] J. S. Pettersson, S. Fischer-Hübner, N. Danielsson, J. Nilsson, M. Bergmann, S. Clauss, T. Kriegelstein, and H. Krasemann: *Making PRIME Usable*. In *Proceedings of the 1st Symposium on Usable Privacy and Security (SOUPS)*, pages 53–64, 2005.
- [PHGo7] E. Paulos, R. Honicky, and E. Goodman: *Sensing Atmosphere*. In *Proceedings of the Workshop on Sensing on Everyday Mobile Phones in Support of Participatory Research (SenSys Workshop)*, pages 15–16, 2007.
- [PSo8] F. Pingel and S. Steinbrecher: *Multilateral Secure Cross-Community Reputation Systems for Internet Communities*. In S. Furnell, S. Katsikas, and A. Lioy (editors): *Trust, Privacy and Security in Digital Business*, volume 5185 of *Lecture Notes in Computer Science*, pages 69–78. Springer Berlin Heidelberg, 2008.
- [RCK<sup>+</sup>10] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu: *Ear-Phone: An End-to-end Participatory Urban Noise Mapping System*. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 105–116, 2010.
- [RGC10] K. Renaud and D. Gálvez-Cruz: *Privacy: Aspects, Definitions and a Multi-faceted Privacy Preservation Approach*. In *Proceedings of the 2010 Information Security for South Africa Conference (ISSA)*, pages 1–8, 2010.
- [RH07] M. Raya and J.-P. Hubaux: *Securing Vehicular Ad Hoc Networks*. *Journal of Computer Security*, 15(1):39–68, 2007.
- [RPH<sup>+</sup>07] S. Reddy, A. Parker, J. Hyman, J. A. Burke, D. Estrin, and M. Hansen: *Image Browsing, Processing, and Clustering for Participatory Sensing: Lessons from a DietSense Prototype*. In *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets)*, pages 13–17, 2007.
- [RSA78] R. L. Rivest, A. Shamir, and L. Adleman: *A Method for Obtaining Digital Signatures and Public-key Cryptosystems*. *Communications of the ACM*, 21(2):120–126, 1978.
- [RSB<sup>+</sup>09] S. Reddy, V. Samanta, J. A. Burke, D. Estrin, M. Hansen, and M. B. Srivastava: *MobiSense – Mobile Network Services for Coordinated Participatory Sensing*. In *Proceedings of the International Symposium on Autonomous Decentralized Systems (ISADS)*, pages 1–6, 2009.

- [SBE<sup>+</sup>08] K. Shilton, J. A. Burke, D. Estrin, M. Hansen, and M. B. Srivastava: *Participatory Privacy in Urban Sensing*. In *Proceedings of the International Workshop on Mobile Devices and Urban Sensing (MODUS)*, pages 1–7, 2008.
- [SBS<sup>+</sup>11] I. Schweizer, R. Bärthel, A. Schulz, F. Probst, and M. Mühlhäuser: *NoiseMap - Real-time Participatory Noise Maps*. In *Proceedings of the 2nd International Workshop on Sensing Applications on Mobile Phones (PhoneSense)*, pages 1–5, 2011.
- [SCP<sup>+</sup>10] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos: *AnonySense: A System for Anonymous Opportunistic Sensing*. *Journal of Pervasive and Mobile Computing*, 7(1):16–30, 2010.
- [SDAB08] E. P. Stuntebeck, J. S. Davis, II, G. D. Abowd, and M. Blount: *HealthSense: Classification of Health-related Sensor Data through User-assisted Machine Learning*. In *Proceedings of the 9th Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 1–5, 2008.
- [Shio9] K. Shilton: *Four Billion Little Brothers?: Privacy, Mobile Phones, and Ubiquitous Data Collection*. *Communications of the ACM*, 52(11):48–53, 2009.
- [SLB<sup>+</sup>03] B. N. Schilit, A. LaMarca, G. Borriello, W. G. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson: *Challenge: Ubiquitous Location-aware Computing and the "Place Lab" Initiative*. In *Proceedings of the 1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH)*, pages 29–35, 2003.
- [SLZ03] S. Shekhar, C.-T. Lu, and P. Zhang: *A Unified Approach to Spatial Outlier Detection*. *GeoInformatica*, 7(2):139–166, 2003.
- [SMBDF06] A. Solanas, A. Martinez-Balleste, and J. Domingo-Ferrer: *V-MDAV: A Multivariate Microaggregation with Variable Group Size*. In *Proceedings of the 17th IASC Symposium on Computational Statistics (COMPSTAT)*, pages 917–925, 2006.
- [SSF<sup>+</sup>03] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. L. Wong: *SenSay: A Context-aware Mobile Phone*. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers (ISWC)*, pages 248–249, 2003.
- [Ste09] S. Steinbrecher: *Enhancing Multilateral Security in and by Reputation Systems*. In V. Matyáš, S. Fischer-Hübner, D. Cvrček, and P. Švenda (editors): *The Future of Identity in the Information Society*, volume 298 of *IFIP Advances in Information and Communication Technology*, pages 135–150. Springer Berlin Heidelberg, 2009.
- [Swe02] L. Sweeney: *K-anonymity: A Model for Protecting Privacy*. *International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems*, 10(5):557–570, 2002.
- [SZLZ10] J. Shi, R. Zhang, Y. Liu, and Y. Zhang: *PriSense: Privacy-preserving Data Aggregation in People-centric Urban Sensing Systems*. In *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9, 2010.
- [TBGE10] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson: *Cooperative Transit Tracking using Smart-phones*. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 85–98, 2010.
- [TLA10] S. Trifunovic, F. Legendre, and C. Anastasiades: *Social Trust in Opportunistic Networks*. In *Proceedings of the 29th IEEE Conference on Computer Communications (INFOCOM)*, pages 1–6, 2010.

- [TLH<sup>+</sup>10] K. P. Tang, J. Lin, J. I. Hong, D. P. Siewiorek, and N. Sadeh: *Rethinking Location Sharing: Exploring the Implications of Social-driven vs. Purpose-driven Location Sharing*. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (UbiComp)*, pages 85–94, 2010.
- [TRL<sup>+</sup>09] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson: *VTrack: Accurate, Energy-aware Road Traffic Delay Estimation using Mobile Phones*. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 85–98, 2009.
- [vKSW11] M. von Kaenel, P. Sommer, and R. Wattenhofer: *Ikarus: Large-Scale Participatory Sensing at High Altitudes*. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 55–60, 2011.
- [Wes67] A. F. Westin: *Privacy and Freedom*. Atheneum, New York, 1967.
- [WH09] Y. Wei and Y. He: *A Pseudonym Changing-based Anonymity Protocol for P2P Reputation Systems*. In *Proceedings of the 1st International Workshop on Education Technology and Computer Science (ETCS)*, pages 975–980, 2009.
- [YZR11] H. Yang, J. Zhang, and P. Roe: *Using Reputation Management in Participatory Sensing for Data Classification*. *Procedia Computer Science*, 5(0):190–197, 2011.
- [ZLC<sup>+</sup>08] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma: *Understanding Mobility based on GPS Data*. In *Proceedings of the 10th ACM International Conference on Ubiquitous Computing (UbiComp)*, pages 312–321, 2008.
- [ZLK10] Z. Zhang, J. Liu, and Y. Kadobayashi: *STARS: A Simple and Efficient Scheme for Providing Transparent Traceability and Anonymity to Reputation Systems*. In *Proceedings of the International Workshop on Autonomous and Spontaneous Security (SETOP)*, pages 170–187, 2010.





## LIST OF ACRONYMS

---

3G	3rd Generation
CCTV	Closed-Circuit Television
FN	False Negatives
FP	False Positives
GIS	Geographic Information System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
ID	Identifier
IP	Internet Protocol
LOF	Local Outlier Factor
MCC	Matthews Correlation Coefficient
MDAV	Maximum Distance to Average Vector
NFC	Near Field Communication
PDV	Personal Data Vault
RPM	Reputation and Pseudonym Manager
RT	Reputation Token
RuP	Reputation using Pseudonyms
SMS	Short Message Service
TCP	Transport Control Protocol
TN	True Negatives
TP	True Positives
V-MDAV	Variable-size Maximum Distance to Average Vector



We first provide background information about blind signatures in Section A.1. In particular, we detail the blind RSA signature scheme utilized in Chapter 7. Based on these foundations, we provide details about the underlying mechanisms used in the generation of pseudonyms (see Section 7.2.1) and the generation of Reputation Tokens (RTs) (see Section 7.2.1) introduced in Sections A.2 and A.3, respectively.

#### A.1 BLIND SIGNATURES, RSA SIGNATURES AND BLIND RSA SIGNATURES

In [Cha83], Chaum proposed the first scheme to create blind signature based on a public-key cryptographic system with signers and providers. The signer has a secret signing function  $s'$  and its public inverse  $s$ , such that  $s(s'(m)) = m$  for a message  $m$ . The provider has two secret cryptographic functions  $c$  and  $c'$ , where  $c'$  is the inverse of  $c$ . Hence, the provider can send a ciphertext  $c(m)$  to the signer, which returns  $s'(c(m))$  to the provider. The provider can then obtain the message  $s'(m)$  signed by the supplier using the function  $c'$ , such that  $c'(s'(c(m))) = s'(m)$ . Anyone can then verify the signature on message  $m$  by checking that  $s(s'(m)) = m$ . Hence, the supplier signed the message  $m$  without knowing  $m$ , i.e., a blind signature.

The RSA [RSA78] signature scheme can be used for implementing a blind signature scheme [Gen08]. In RSA, the provider signs its own messages, i.e., the provider and the signer are the same entity. A signature  $s$  of message  $m$  is  $s = m^e \bmod n$ , where  $(e, n)$  is the *signature key*,  $n = p \cdot q$  and  $p$  and  $q$  are two arbitrarily chosen large prime numbers. The parameter  $e$ , ( $e < n$ ), is a relative prime to the totient of  $n$ ,  $\phi(n) = (p-1)(q-1)$ . For checking the correctness of an RSA signature, the verifier needs a *verification key*  $(d, n)$ , where  $d$  is the inverse of  $e \bmod \phi(n)$ , i.e.,  $e \cdot d \bmod \phi(n) \equiv 1$ . The verification of signature is done by checking if  $s^d$  is equivalent to  $m \bmod n$ .

In a blind RSA signature scheme, the provider and the signer are different entities. The signer is a third party that does not know the content of the message to be signed. The blind RSA signature scheme incorporates a blinding factor  $r$  to the RSA signature scheme and works as follows. The provider of the message  $m$ , which is to be blind signed, selects a random blinding factor  $r \bmod n$  and generates a blinded message  $m' = m \cdot r^e \bmod n$ . The blinding factor  $r$  is a secret only known to the provider and  $(e, n)$  is publicly known. The blinded message  $m'$  is sent to the signer, who returns the signature  $s' = (m')^d \bmod n$  to the provider. The signature  $s'$  on message  $m'$  is obtained using the signature key  $(d, n)$ , which is a secret only known to the signer. The provider can then calculate the signature  $s = m^d \bmod n$  on message  $m$  as

$$s' = (m')^d \bmod n = (m \cdot r^e)^d \bmod n = m^d \cdot r^{e \cdot d} \bmod n = m^d \cdot r \bmod n \quad (14)$$

The provider then divides  $s'$  by  $r$  to obtain the signature  $s$ , i.e.,  $s = s'/r$ . Anyone can verify the signature  $s$  on message  $m$  by checking if  $s^e$  is equivalent to  $m \bmod n$ .

#### A.2 GENERATION OF PSEUDONYMS USING RSA SIGNATURES

We assume that each client has a permanent identifier ID, a private key  $PR(d_{\text{client}}, n_{\text{client}})$ , a public key  $PU(e_{\text{client}}, n_{\text{client}})$ , and it is registered with the RPM. The blind RSA signature

scheme, which is presented in Section A.1, is the blind signature scheme used in the rest of this appendix. For each  $T$ , the RPM generates a new private/public pair of *signature keys* common to all clients:  $\text{PR}(d_{\text{signature}}, n_{\text{signature}})$  and  $\text{PU}(e_{\text{signature}}, n_{\text{signature}})$ . The client first generates a private/public key pair  $\text{PR}(d_p, n_p)$  and  $\text{PU}(e_p, n_p)$  for its new pseudonym. The client uses  $n_p$  as its new pseudonym referred to as  $P$  and generates the corresponding signature  $s_p$  as follows. The client first prepares the message  $m_p$  using  $n_p$ , the RPM's public *signature key* for the time period  $T$ ,  $\text{PU}(e_{\text{signature}}, n_{\text{signature}})$ , and the blinding factor  $r$  modulo  $n_p$ , as follows:

$$m_p = n_p \cdot r^{e_{\text{signature}}} \bmod n_{\text{signature}} \quad (15)$$

The client creates a signature  $s_{m_p}$  using a function of the concatenation  $f$  of the triplet  $(m_p, \text{ID}, T)$  signed with its permanent private key to guarantee the authenticity of  $m_p$ :

$$s_{m_p} = f(m_p \parallel \text{ID} \parallel T)^{d_{\text{client}}} \bmod n_{\text{client}} \quad (16)$$

The client transmits  $m_p$ ,  $s_{m_p}$ , its ID, and the time interval of validity  $T$  for  $P$  to the RPM for blind signature. The RPM verifies the authenticity of  $m_p$  and that the client has no existing pseudonym for this time interval. After verification, the RPM generates the blind signature  $s_{\text{RPM}}$  signing  $m_p$ :

$$s_{\text{RPM}} = m_p^{d_{\text{signature}}} \bmod n_{\text{signature}} \quad (17)$$

The client finally generates the pseudonym's signature  $s_p$  from the blind signature  $s_{\text{RPM}}$  that achieves the generation of  $P$ , which becomes  $P_{\text{current}}$ :

$$s_p = s_{\text{RPM}} \cdot r^{-1} \bmod n_{\text{signature}} \quad (18)$$

### A.3 GENERATION OF REPUTATION TOKENS USING RSA SIGNATURES

We assume that the RPM generates a set of *transfer keys* in the bootstrapping phase. Each *transfer key* pair is associated to a reputation value and determines the reputation associated to a given RT. For each RT, the client selects a random bit string  $\text{ID}_{\text{RT}}$  as identifier and prepares the message  $m_{\text{RT}}$  for blind signature using the public *transfer key* corresponding to the RT's value:

$$m_{\text{RT}} = \text{ID}_{\text{RT}} \cdot r^{e_{\text{transfer}}} \bmod n_{\text{transfer}} \quad (19)$$

The client signs the message  $m_{\text{RT}}$  with the signature  $s_{m_{\text{RT}}}$  using the private key  $\text{PR}(d_{p_{\text{current}}}, n_{p_{\text{current}}})$  associated to  $P_{\text{current}}$ . The real identity of the client is hence not revealed while transferring reputation from one pseudonym to the next.

$$s_{m_{\text{RT}}} = f(m_{\text{RT}} \parallel P_{\text{current}} \parallel R_{\text{score}})^{d_{p_{\text{current}}}} \bmod n_{p_{\text{current}}} \quad (20)$$

The client transmits  $m_{\text{RT}}$ ,  $s_{m_{\text{RT}}}$ ,  $P_{\text{current}}$ , and  $R_{\text{score}}$  to the RPM for blind signature. The RPM verifies that  $m_{\text{RT}}$  is used for the first time as well as the balance of the reputation account of  $P_{\text{current}}$  before decrementing it by  $R_{\text{score}}$ . After verification, the RPM blindly signs  $m_{\text{RT}}$  with the corresponding private *signature key*. The client finally uses the blind signature to generate the final signature of the RT.

CURRICULUM VITÆ

---

## PERSONAL DETAILS

<i>Name</i>	Delphine Christin
<i>Date of Birth</i>	22 February 1985
<i>Place of Birth</i>	Fréjus, France
<i>Nationality</i>	French

## EDUCATION

<i>since 04/2009</i>	Technische Universität Darmstadt, Darmstadt, Germany Doctoral candidate at the Department of Computer Science
<i>10/2007 – 04/2009</i>	Technische Universität Darmstadt, Darmstadt, Germany Studies of Electrical Engineering and Information Technology Degree: Diplom-Ingenieur (Dipl.-Ing.), M.Sc. equivalent
<i>09/2005 – 04/2009</i>	Ecole Nationale Supérieure de l'Electronique et de ses Applications, Cergy, France Studies of Electrical Engineering and Telecommunications Degree: Diplôme d'Ingénieur, M.Sc. equivalent
<i>09/2003 – 06/2005</i>	Lycée Kléber, Strasbourg, France Classes Préparatoire aux Grandes Ecoles, intensive preparatory school for entrance examination of French engineering schools
<i>09/2001 – 06/2003</i>	Lycée Scheurer Kestner, Thann, France Degree: Diplôme du Baccalauréat Général, high school diploma

## WORK EXPERIENCE

<i>since 10/2009</i>	Technische Universität Darmstadt, Darmstadt, Germany Research associate at the Secure Mobile Networking Lab
<i>since 04/2009</i>	Technische Universität Darmstadt, Darmstadt, Germany Research associate at the Center for Advanced Security Research Darmstadt
<i>06/2007 – 08/2007</i>	Illinois Institute of Technology, Chicago, United States of America Internship in designing power systems for emergency trucks
<i>06/2006 – 07/2006</i>	Endress+Hauser, Maulburg, Germany Internship in software development for mobile devices

## TEACHING ACTIVITIES

<i>since 04/2009</i>	Technische Universität Darmstadt, Darmstadt, Germany Teaching assistant for the lecture "Mobile Networking"
<i>since 04/2009</i>	Technische Universität Darmstadt, Darmstadt, Germany Coordinator of the seminar "Advanced Topics in Mobile Networking"
<i>since 04/2009</i>	Technische Universität Darmstadt, Darmstadt, Germany Tutor for various Diploma, Bachelor, and Master theses as well as seminars, lab exercises, and projects

Darmstadt, 22 January 2013





## AUTHOR'S PUBLICATIONS

## PRIMARY AUTHOR

*Journals and Book Chapters*

1. D. Christin, C. Roßkopf, M. Hollick, L. A. Martucci, and S. S. Kanhere. IncogniSense: An Anonymity-preserving Reputation Framework for Participatory Sensing Applications. *Accepted for publication in Pervasive and Mobile Computing*, 2013.
2. D. Christin and M. Hollick. Roadmap for Privacy Protection in Mobile Sensing Applications. In Serge Gutwirth, Ronald Leenes, Paul de Hert, and Yves Pouillet, editors, *European Data Protection: Coming of Age*, pages 203–222. Springer Netherlands, 2013.
3. D. Christin, P. Sánchez López, A. Reinhardt, M. Hollick, and M. Kauer. Share with Strangers: Privacy Bubbles as User-Centered Privacy Control for Mobile Content Sharing Applications. *Information Security Technical Report*, 17(3):105–116, 2013.
4. D. Christin, C. Roßkopf, and M. Hollick. uSafe: A Privacy-aware and Participative Mobile Application for Citizen Safety in Urban Environments. *Accepted for publication in Pervasive and Mobile Computing*, 2012.
5. D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick. A Survey on Privacy in Mobile Participatory Sensing Applications. *Journal of Systems and Software*, 84(11):1928–1946, 2011.
6. D. Christin, P. S. Mogre, and M. Hollick. Survey on Wireless Sensor Network Technologies for Industrial Automation: The Security and Quality of Service Perspectives. *Future Internet*, 2(2):96–125, 2010.

*Conferences and Workshops*

7. D. Christin, A. Reinhardt, M. Hollick, and K. Trumpold. Exploring User Preferences for Privacy Interfaces in Mobile Sensing Applications. In *Proceedings of 11th ACM International Conference on Mobile and Ubiquitous Multimedia (MUM)*, pages 14:1–14:10, 2012.
8. D. Christin, C. Büttner, and N. Repp. CachedSensing: Exploring and Documenting the Environment as a Treasure Hunt. In *Proceedings of the 7th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 977–985, 2012.
9. D. Christin, P. Sánchez López, A. Reinhardt, M. Hollick, and M. Kauer. Privacy Bubbles: User-Centered Privacy Control for Mobile Content Sharing Applications. In *Proceedings of the 6th Workshop on Information Security Theory and Practice (WISTP)*, pages 71–86, 2012.
10. D. Christin, A. Bentolila, and M. Hollick. Friend is Calling: Exploiting Mobile Phone Data to Help Users in Setting their Privacy Preferences. In *Proceedings of the 4th International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use (IWSSI/SPMU)*, pages 1–7, 2012.
11. D. Christin, C. Roßkopf, M. Hollick, L. A. Martucci, and S. S. Kanhere. IncogniSense: An Anonymity-preserving Reputation Framework for Participatory Sensing Applications. In *Proceedings of the 10th IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 135–143, 2012.

12. D. Christin, J. Guillemet, A. Reinhardt, M. Hollick, and S. S. Kanhere. Privacy-preserving Collaborative Path Hiding for Participatory Sensing Applications. In *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 341–350, 2011.
13. D. Christin, T. Freudenreich, and M. Hollick. A Picture is Worth a Thousand Words: Privacy-aware and Intuitive Relationship Establishment in Online Social Networks. In *Proceedings of the 3rd International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use (IWSSI/SPMU)*, pages 1–6, 2011.

#### *Extended Abstracts and Technical Reports*

14. D. Christin, D. Rodriguez Pons-Sorolla, S. S. Kanhere, and M. Hollick. TrustMeter: A Trust Assessment Framework for Collaborative Path Hiding in Participatory Sensing Applications. Technical Report TR-SEEMOO-2012-02, Technische Universität Darmstadt, October 2012.
15. D. Christin and M. Hollick. We Must Move – We Will Move: On Mobile Phones as Sensing Platforms. In *Proceedings of the 10th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 25–28, 2011.
16. D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick. Fine-grained Access Control Enabling Privacy Support in Wireless Sensor Networks. In *Proceedings of the 9th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 29–32, 2010.
17. D. Christin. Impenetrable Obscurity vs. Informed Decisions: Privacy Solutions for Participatory Sensing. In *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications (PerCom Workshop)*, pages 847–848, 2010.
18. D. Christin, A. Reinhardt, P. S. Mogre, and R. Steinmetz. Wireless Sensor Networks and the Internet of Things: Selected Challenges. In *Proceedings of the 8th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 31–33, 2009.

#### CO-AUTHOR

#### *Conferences and Workshops*

19. A. Reinhardt, D. Christin, and R. Steinmetz. Pre-Allocating Code Mappings for Energy-Efficient Data Encoding in Wireless Sensor Networks. In *Proceedings of the 9th IEEE Workshop on Sensor Networks and Systems for Pervasive Computing (PerCom Workshop)* (accepted for publication), 2013.
20. D. Christin, M. Hollick, and M. Manulis. Security and Privacy Objectives for Sensing Applications in Wireless Community Networks. In *Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6, 2010.
21. A. Reinhardt, D. Christin, M. Hollick, J. Schmitt, P. S. Mogre, and R. Steinmetz. Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks. In *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN)*, pages 33–48, 2010.
22. A. Reinhardt, D. Christin, M. Hollick, and R. Steinmetz. On the Energy Efficiency of Lossless Data Compression in Wireless Sensor Networks. In *Proceedings of the 4th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 873–880, 2009.

*Extended Abstracts*

23. M. Kauer, B. Franz, T. Pfeiffer, M. Heine, and D. Christin. Improving Privacy Settings for Facebook by Using Interpersonal Distance as Criterion. In *Proceedings of the 31st ACM SIGCHI Conference on Human Factors in Computing Systems (CHI Works-in-Progress)* (accepted for publication), 2013.
24. M. Hollick and D. Christin. Fünf Milliarden vernetzte Sensoren: Chancen und Gefahren. In *Proceedings of the 5. Essener Workshop „Neue Herausforderungen in der Netzsicherheit“*, pages 1–2, 2011.



ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

---

Ich versichere hiermit, dass ich die vorliegende Dissertation selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmitteln verfasst habe. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch nicht zu Prüfungszwecken gedient.

*Darmstadt, 22. Januar 2013*

---

Dipl.-Ing. Delphine Christin

